Workshop de Docker y Orquestadores de Containers

Instructor

Diego Chávez

diegochavezcarro@gmail.com

Agenda

- Introducción a Docker.
- Volúmenes en contenedores
- Redes de contenedores
- Docker Compose
- Orquestadores de Containers
- Docker Swarm
- Mesos y Marathon
- Kubernetes

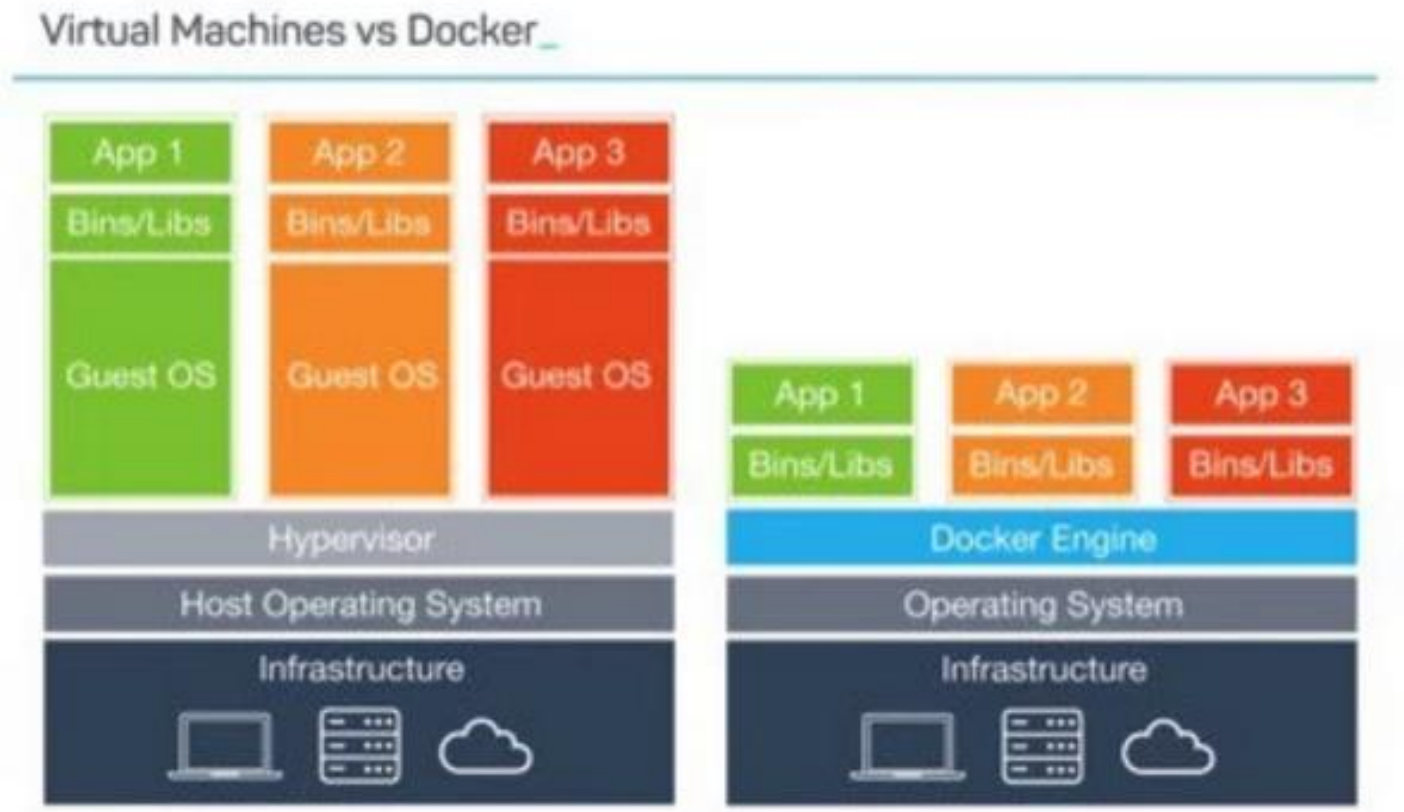
Introducción a Docker

Qué es Docker?

- Es un proyecto de código abierto que permite automatizar el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux. También en Windows, utilizando Windows Containers.

Docker vs VMs

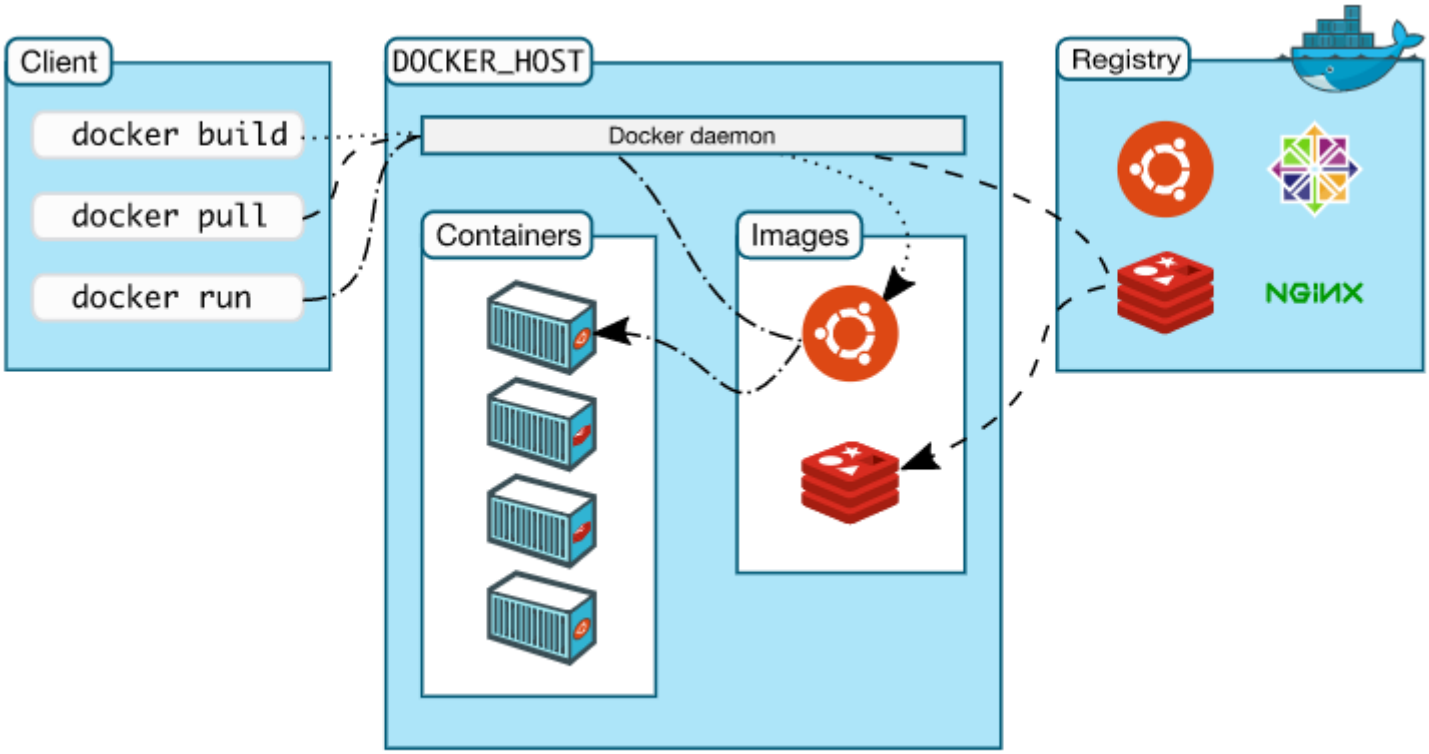
- Más livianos, performantes y “migrables” que las VM tradicionales.
- Mejor Infrastructure as Code



Docker

- Provee la posibilidad de desplegar y ejecutar aplicaciones en un entorno aislado llamado container.
- Este aislamiento seguro permite ejecutar muchos contenedores de forma simultánea en un mismo host.
- Los contenedores son livianos porque no necesitan un hypervisor, se ejecutan directamente en el kernel del host.
- Los containers incluso pueden ejecutarse dentro de una VM.

Arquitectura de Docker



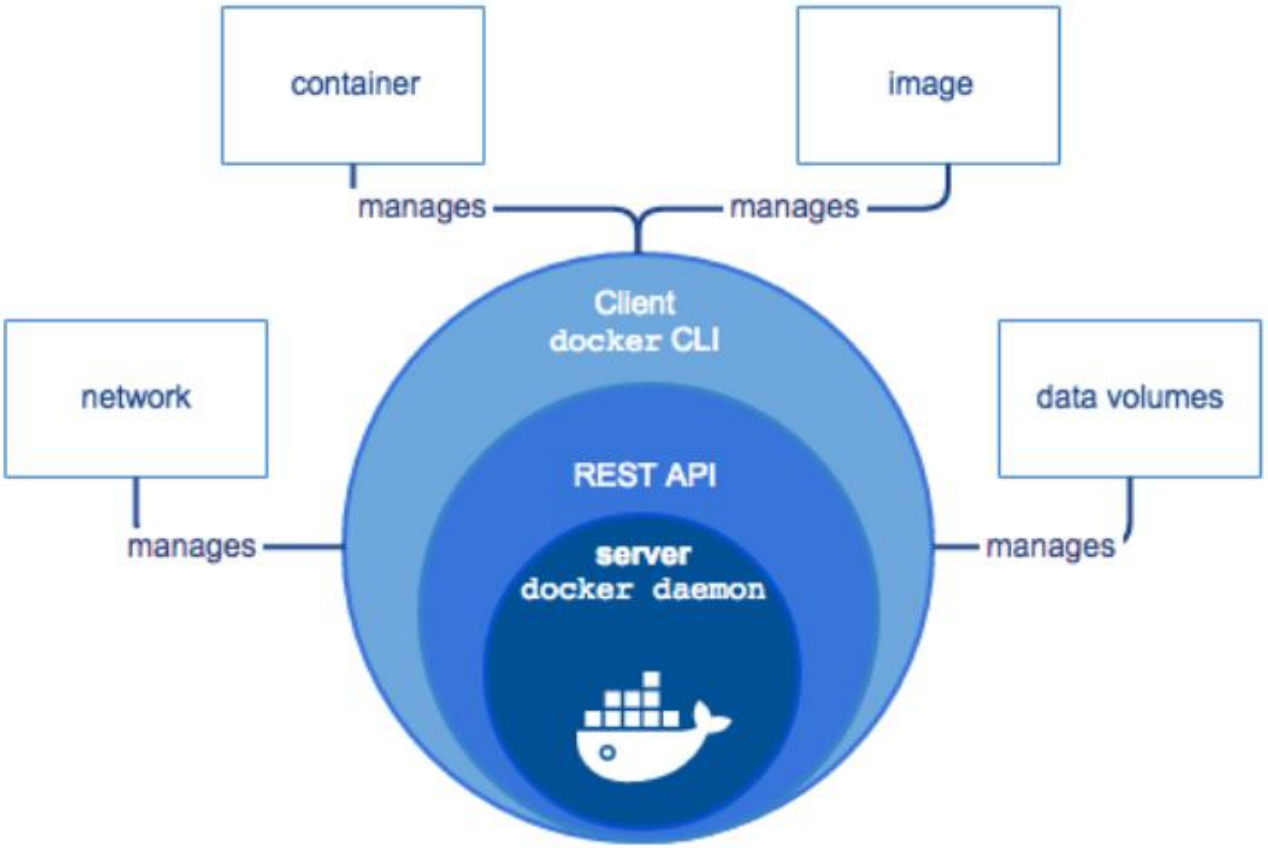
Arquitectura de Docker

- Programado en Lenguaje GO
- Utiliza tecnologías de Linux tales como Namespaces, Control Groups y Union File Systems para aislar los contenedores.
- Docker for Windows: Utiliza Hyper-v para Linux Containers y Windows Containers para correr contenedores nativos de Windows:

https://docs.docker.com/docker-for-windows/

https://github.com/docker/labs/blob/master/windows/windows-containers/README.md

Docker Engine



Instalación

- Linux, por ej Ubuntu:

https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/

- Windows (Docker Toolbox):

https://docs.docker.com/toolbox/toolbox_install_windows/

- Win 10 PRO y Win Server 2016 (Docker for Windows):

https://docs.docker.com/docker-for-windows/install/

- Docker for MAC:
- https://docs.docker.com/docker-for-mac/install/
- Una vez instalado ver las versiones del cliente y del server con “docker -version”

Herramientas Extra

- Docker Toolbox: Docker Engine, Docker Compose,Virtual Box y Docker Machine ya instalados.
- Ubuntu:

https://docs.docker.com/compose/install/#install-compose

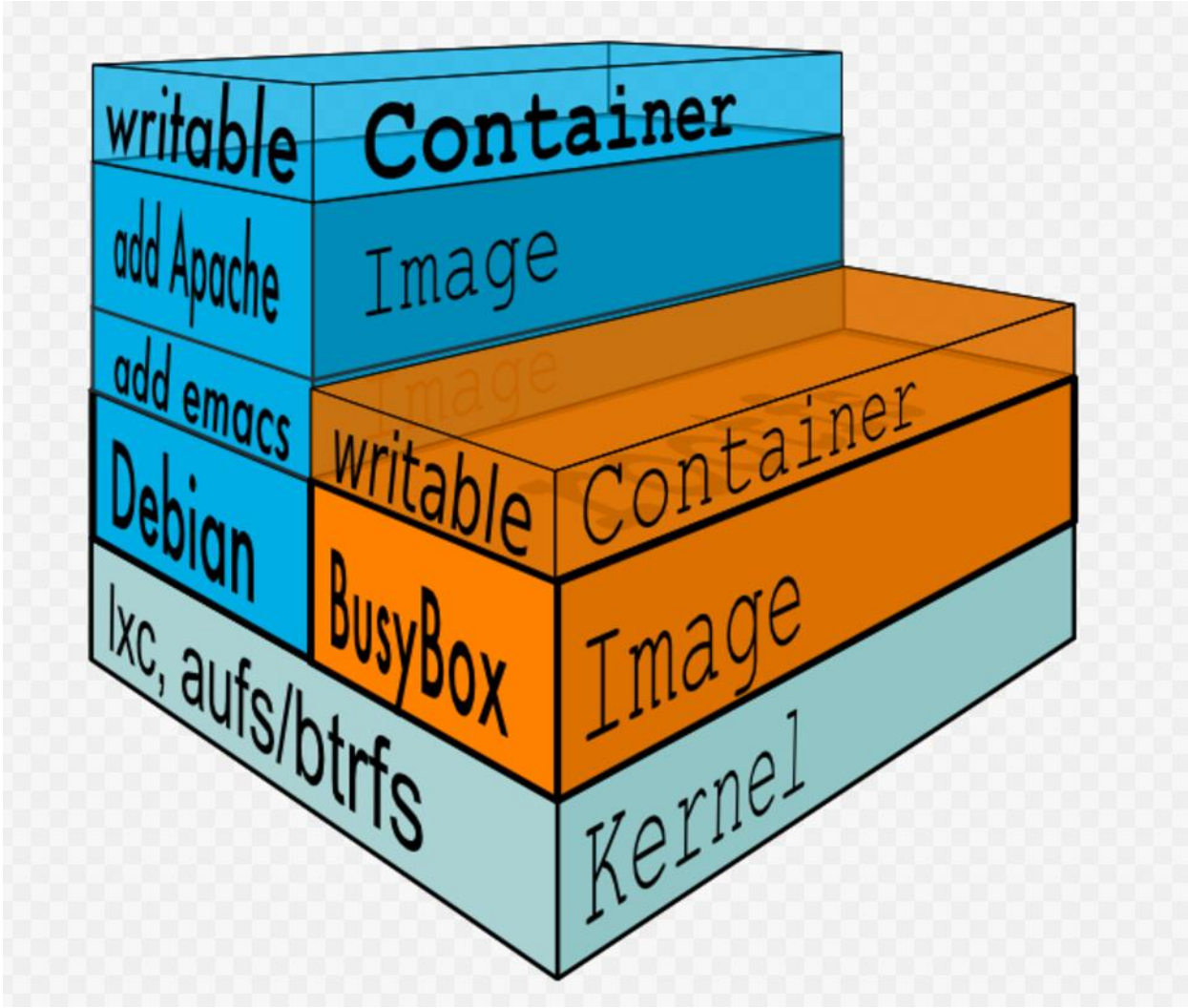
https://www.virtualbox.org/wiki/Linux_Downloads

https://askubuntu.com/questions/367248/how-to-install-virtualbox-from-command-line

https://askubuntu.com/questions/367248/how-to-install-virtualbox-from-command-line/713526#713526

https://docs.docker.com/machine/install-machine/#installing-machine-directly

Imágenes y Containers



Infrastructura como Código

- Concepto importante de DevOps

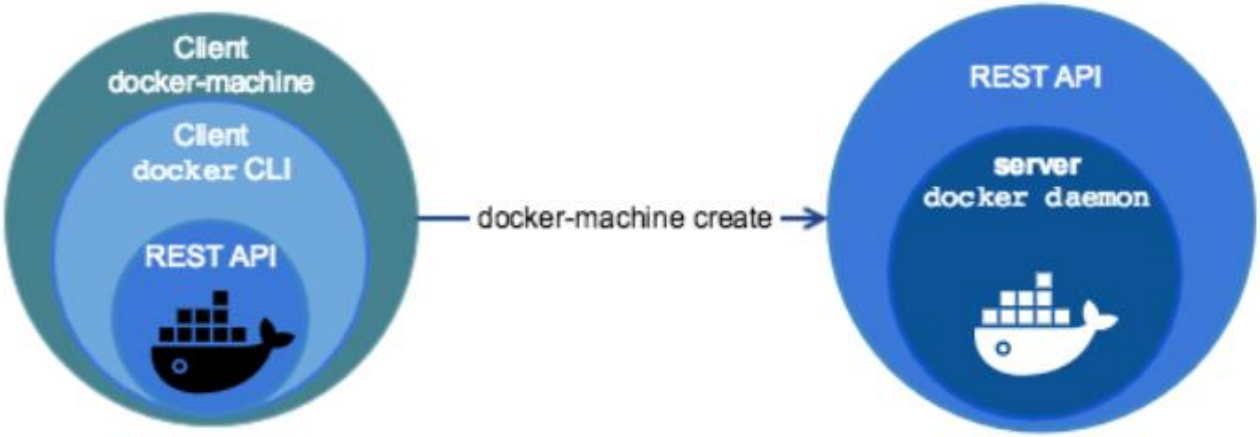
```
FROM openjdk:8-jdk-alpine
RUN  apk update && apk upgrade && apk add netcat-openbsd
RUN mkdir -p /usr/local/organizationservice
ADD @project.build.finalName@.jar /usr/local/organizationservice/
ADD run.sh run.sh
RUN chmod +x run.sh
CMD ./run.sh
```

Carpeta docker_labs:

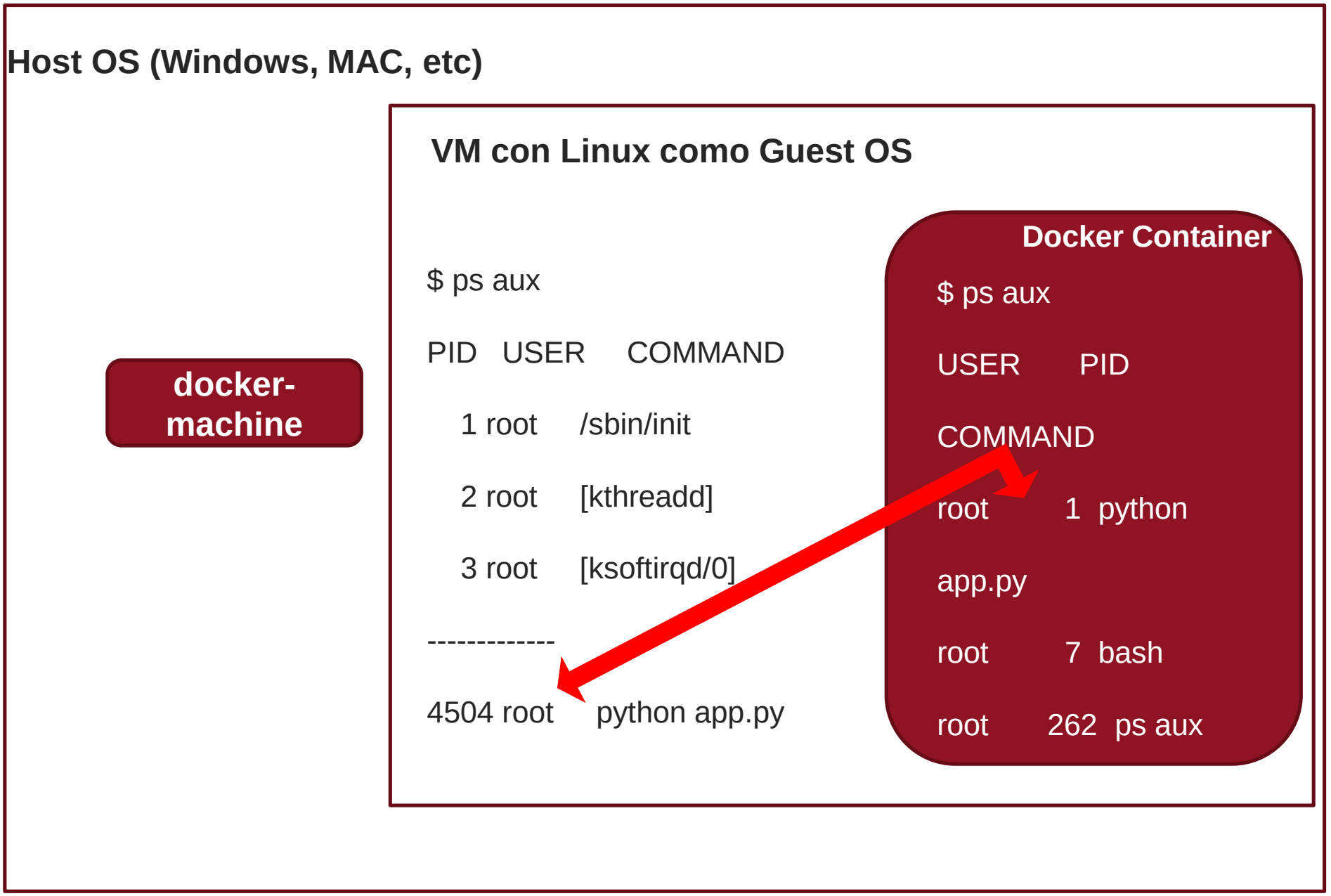
Lab 1: Imágenes y Containers

Lab 2: “Contaneirización” de una aplicación Web

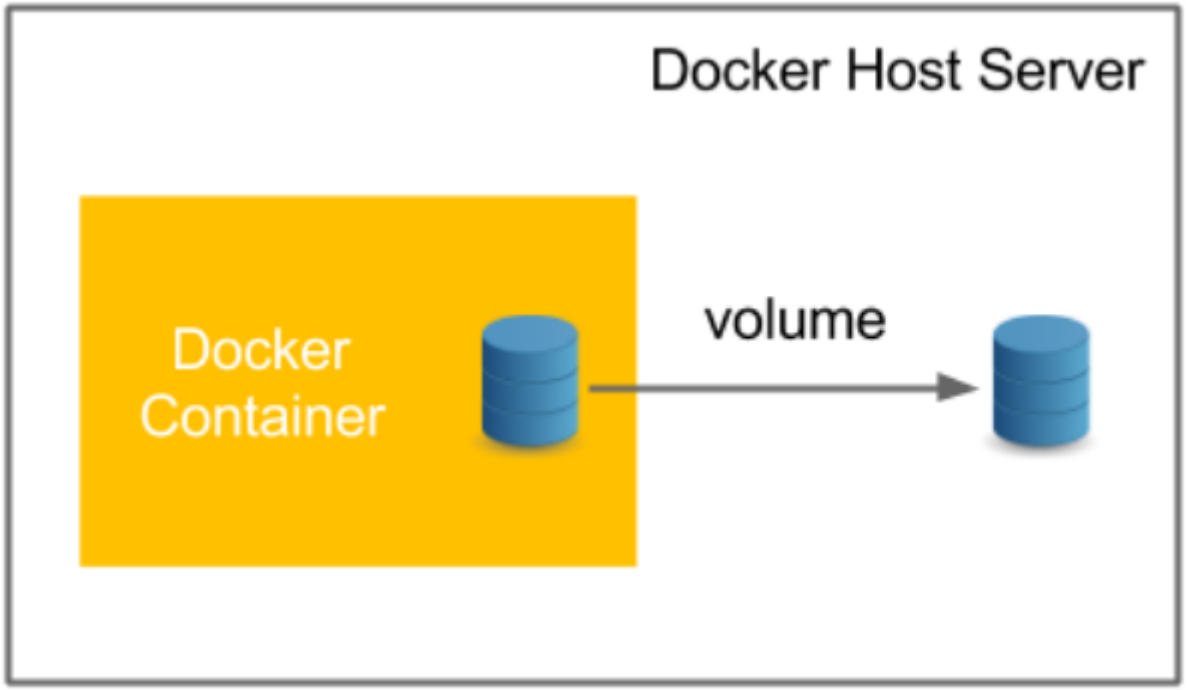
- Uso de docker-machine para apuntar a un docker daemon externo (caso Docker for Windows, Docker Toolbox o Docker for MAC)



Lab 2: Observar Process ID

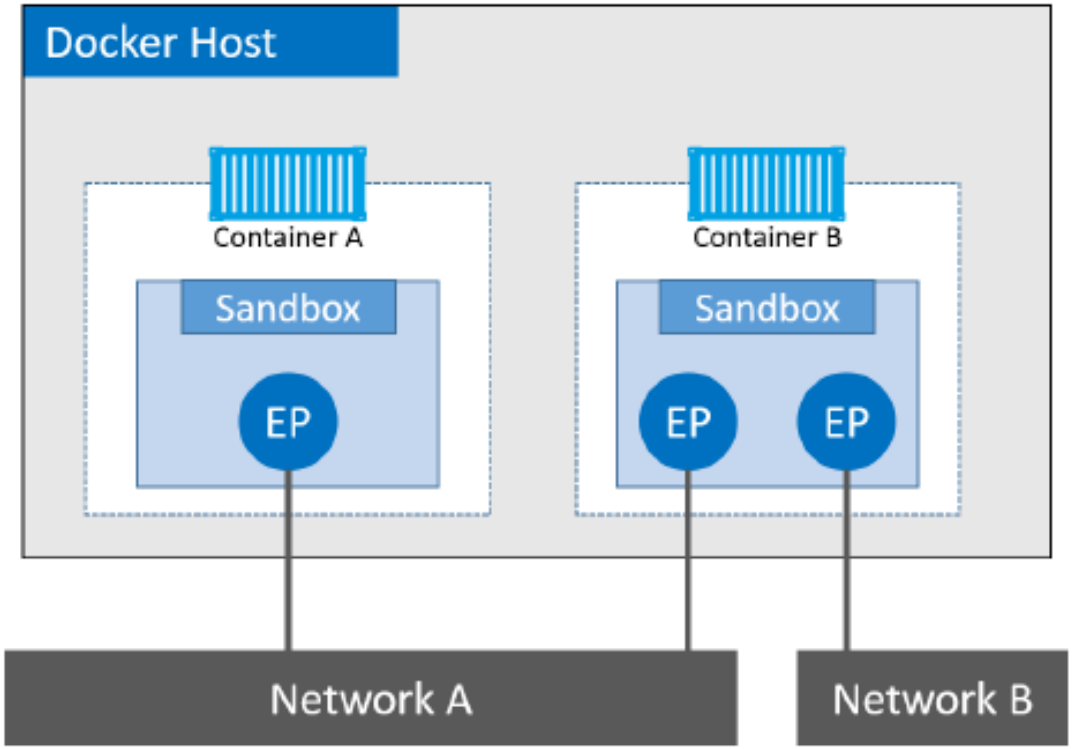


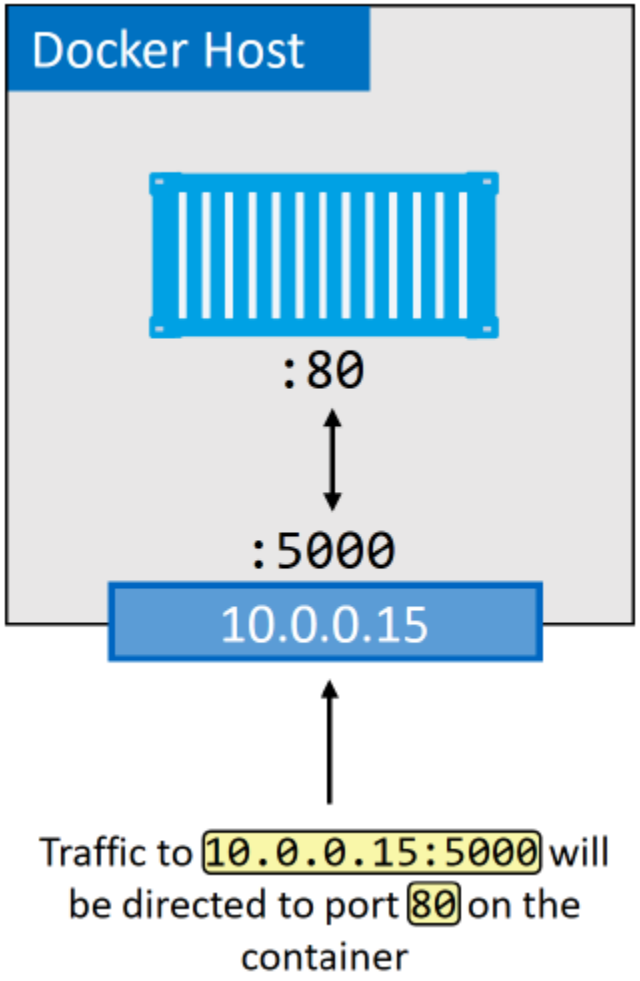
Volúmenes



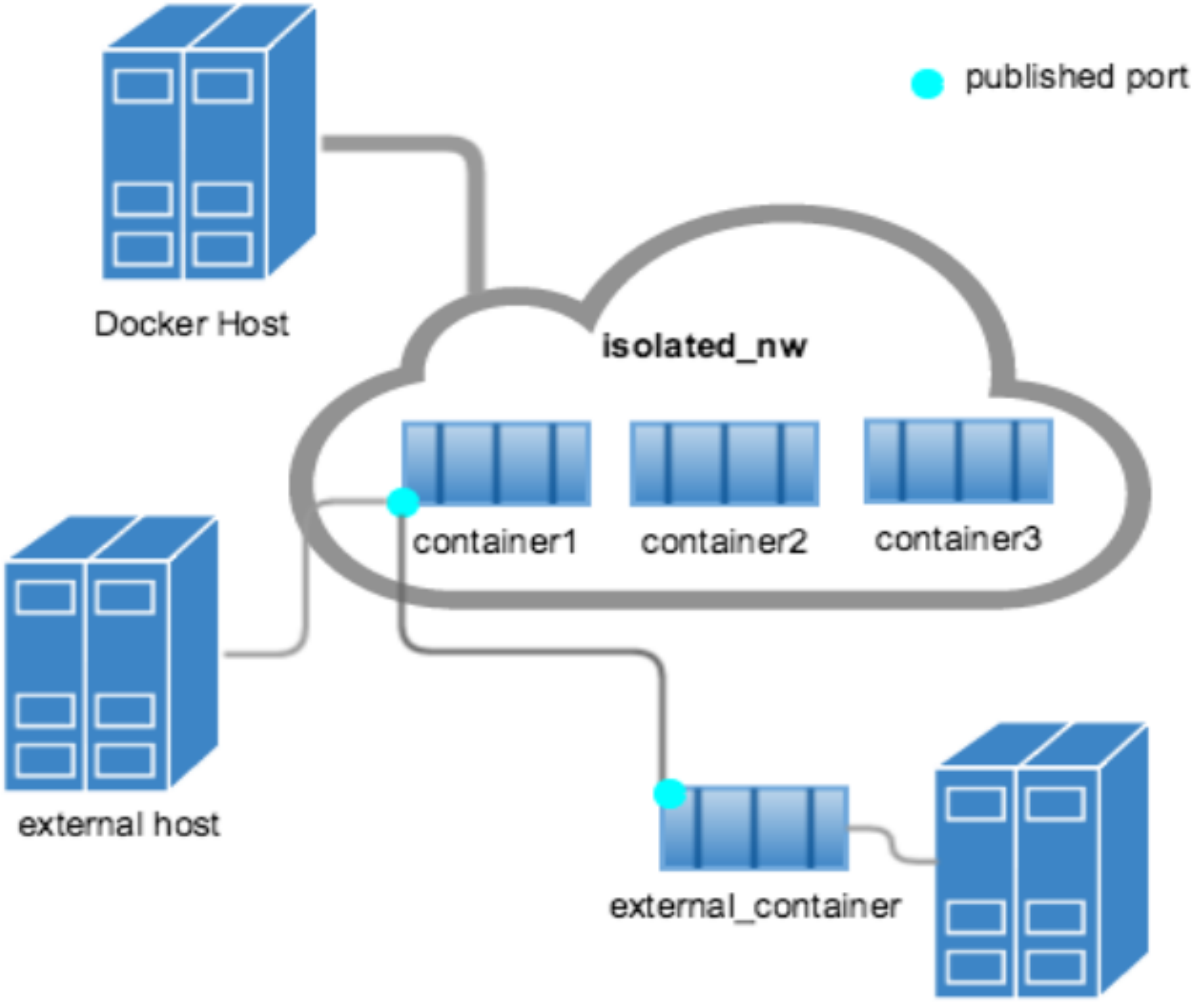
Lab 3: Volúmenes

Networking con Bridge driver

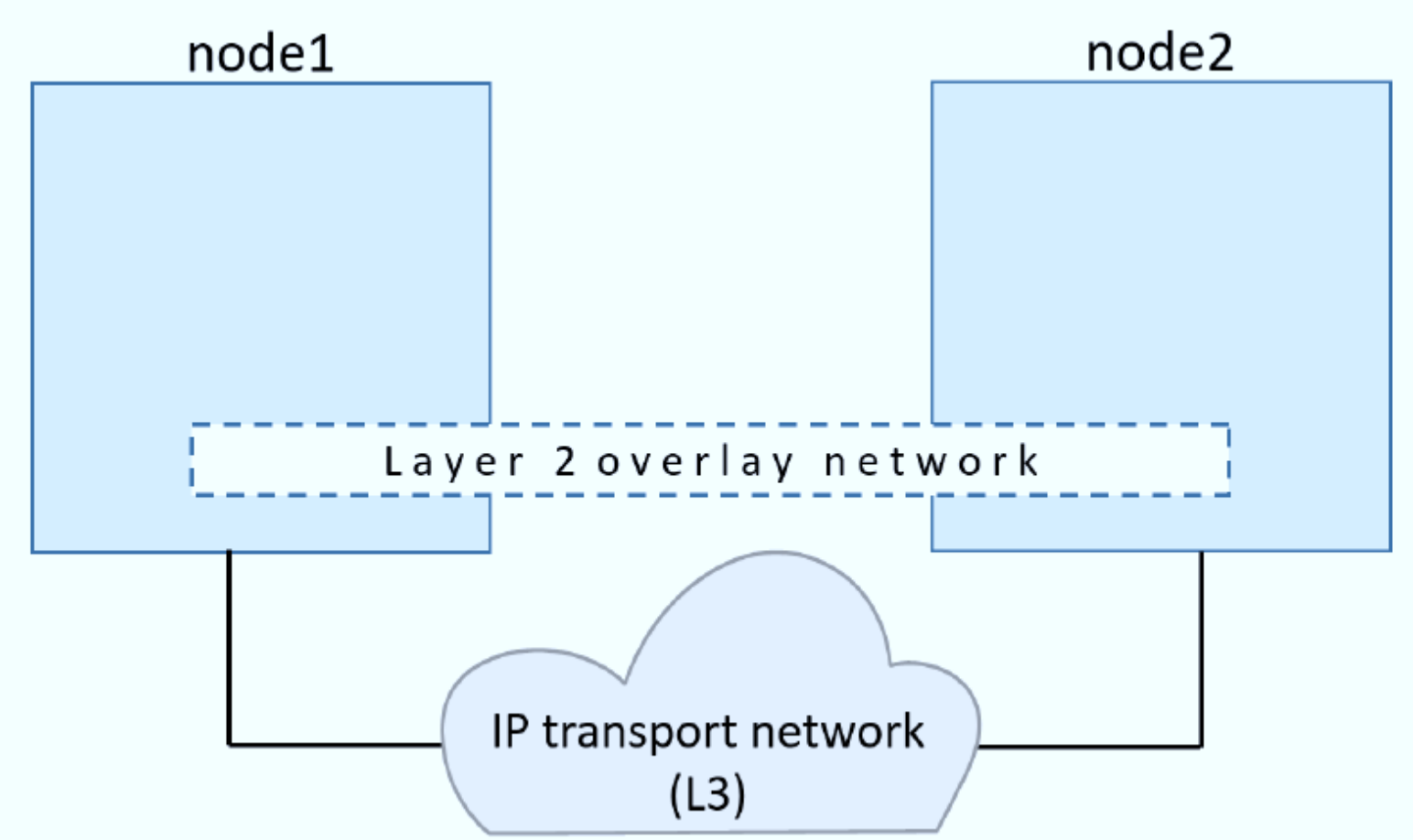




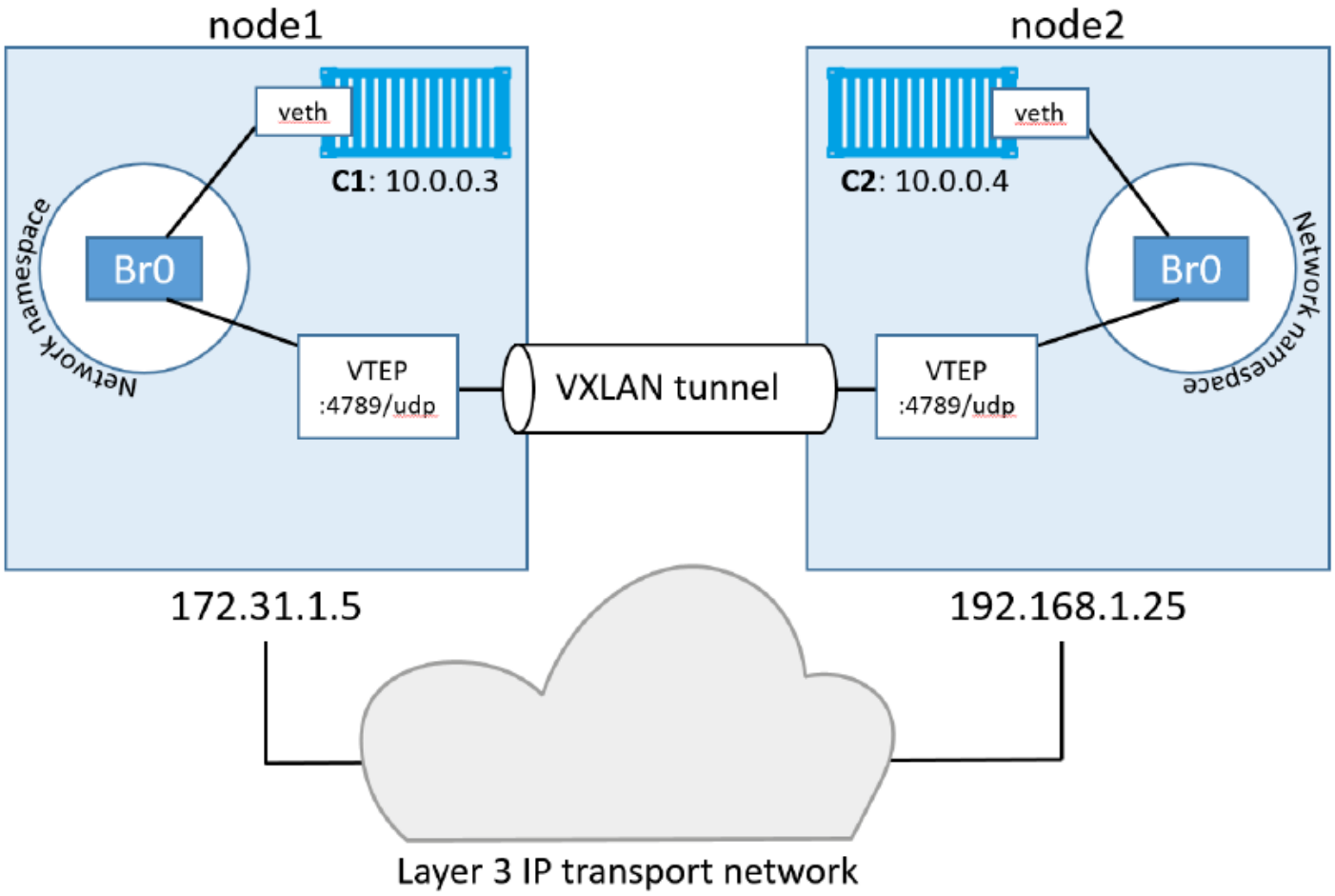
Networking con Bridge driver



Networking con Overlay driver

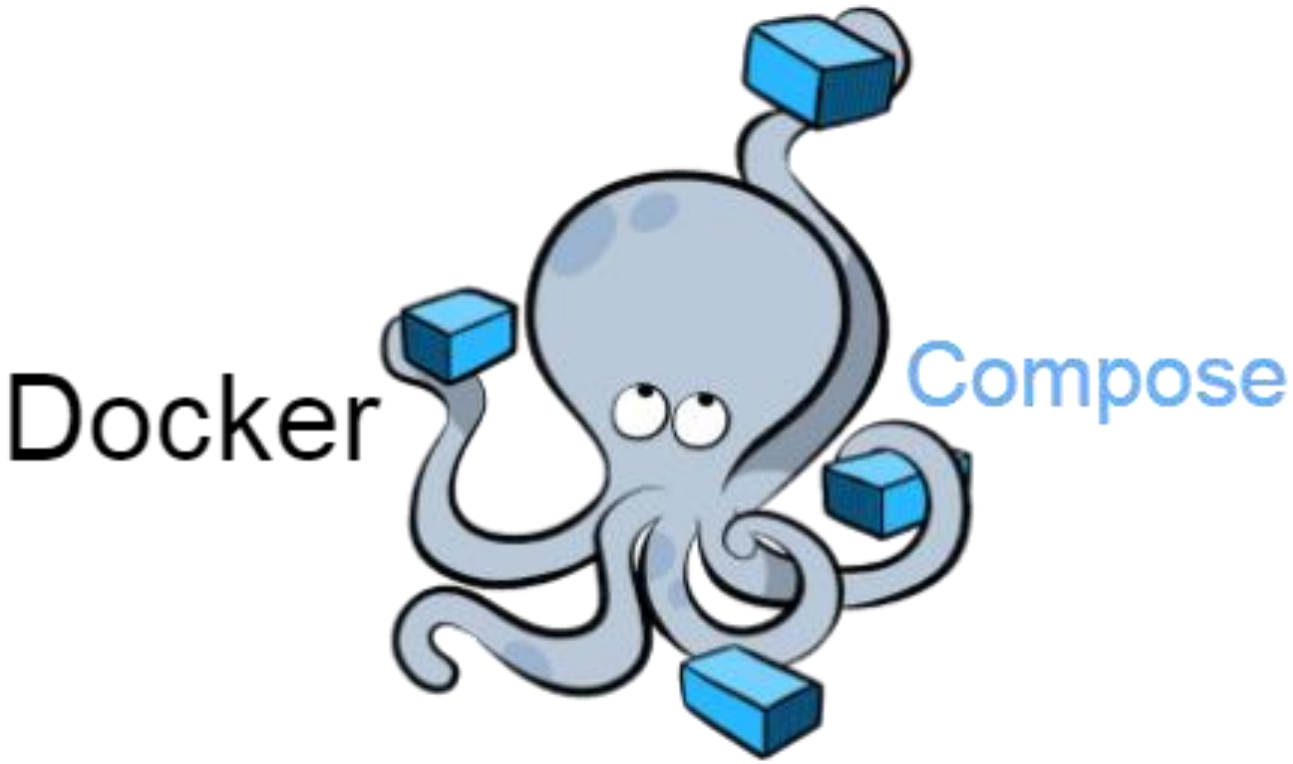


Networking con Overlay driver



Lab 4: Networking

Docker Compose



Docker Compose

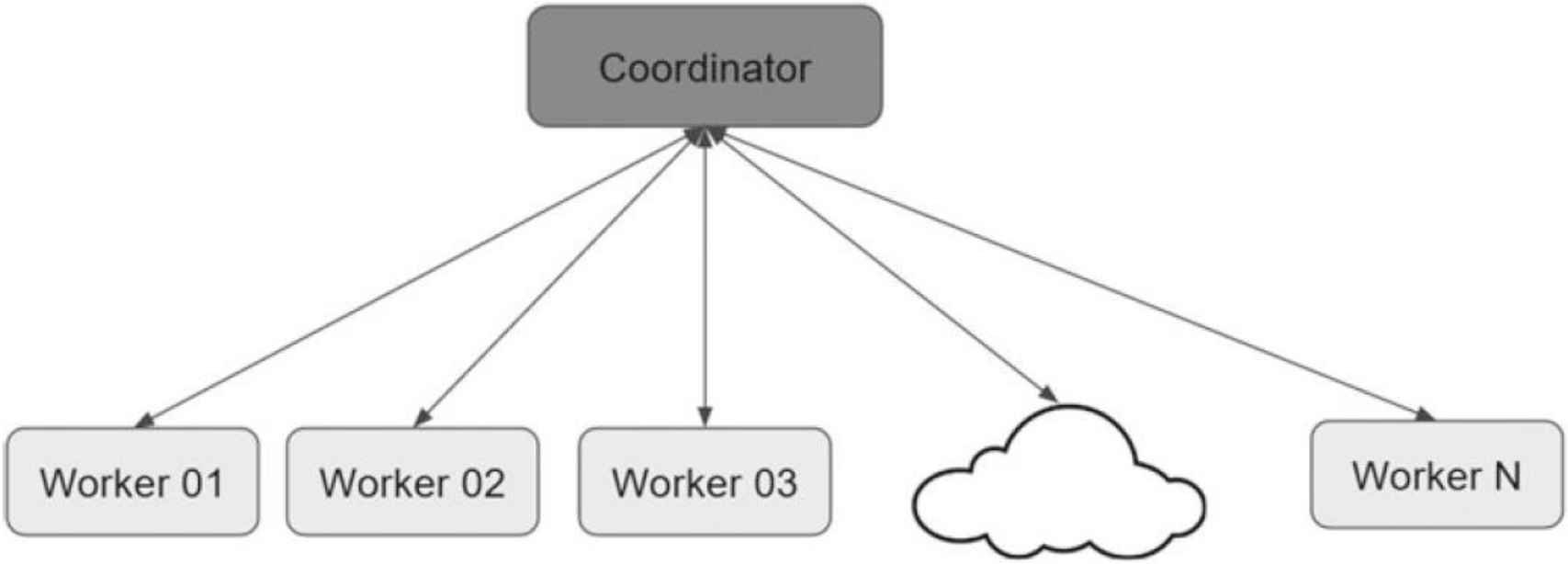
```
version: '3'
services:
  web:
    build: .
    ports:
     - "5000:5000"
  redis:
    image: "redis:alpine"
```

Lab 5: Docker Compose

Orquestadores de Containers

Orquestadores de Containers

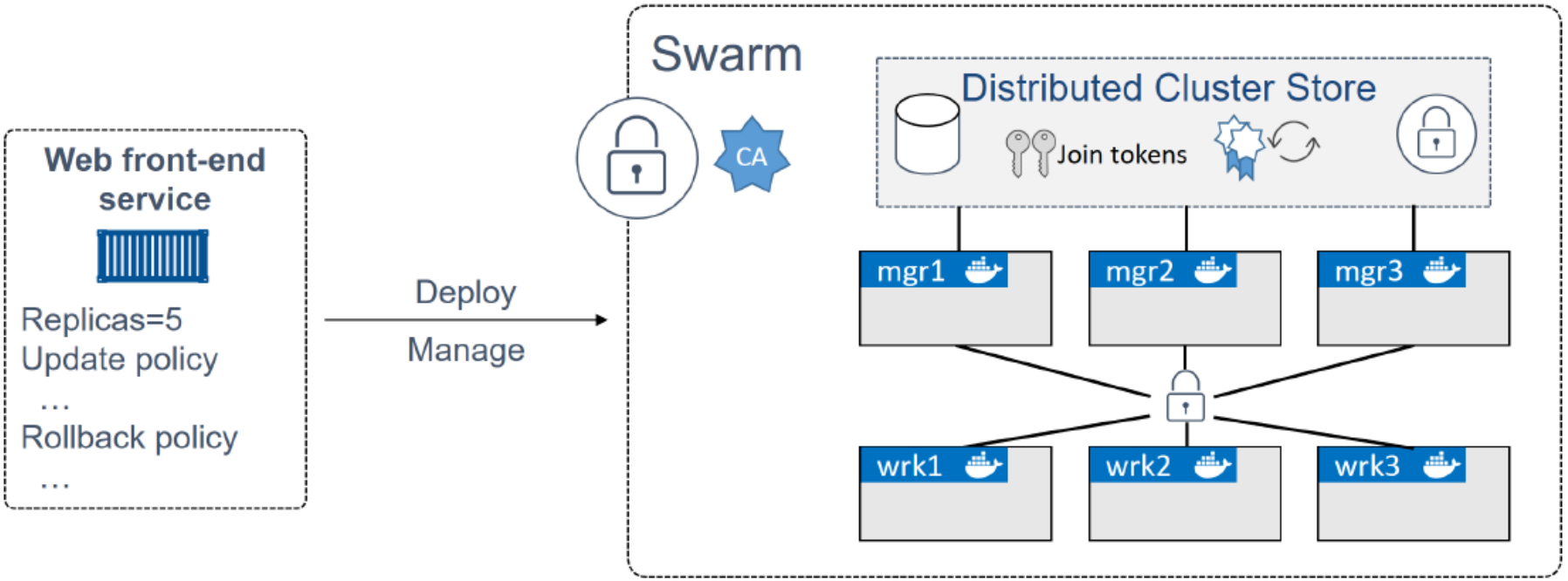
Por lo general tienen además la responsabilidad de Cluster Management.



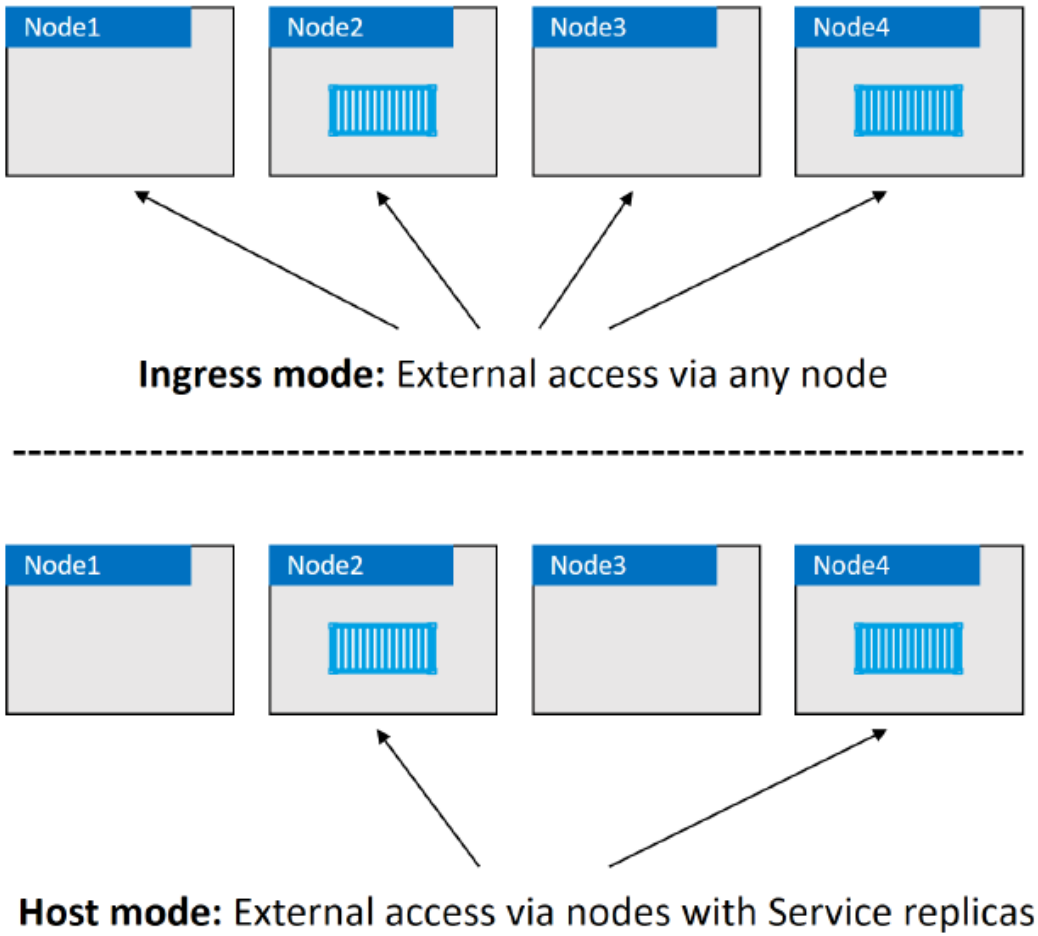
Dashboard de un Cluster Manager


docker
SWARM

Docker Swarm

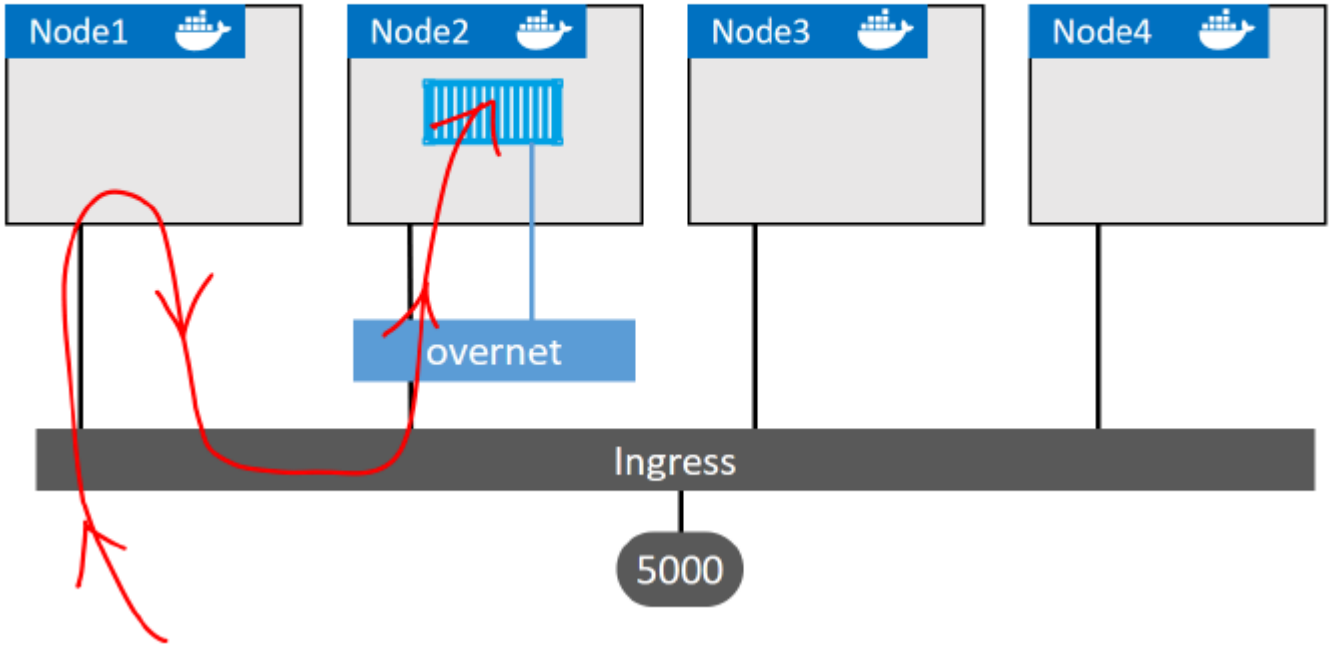


Networking: Modo Ingress vs Modo Host

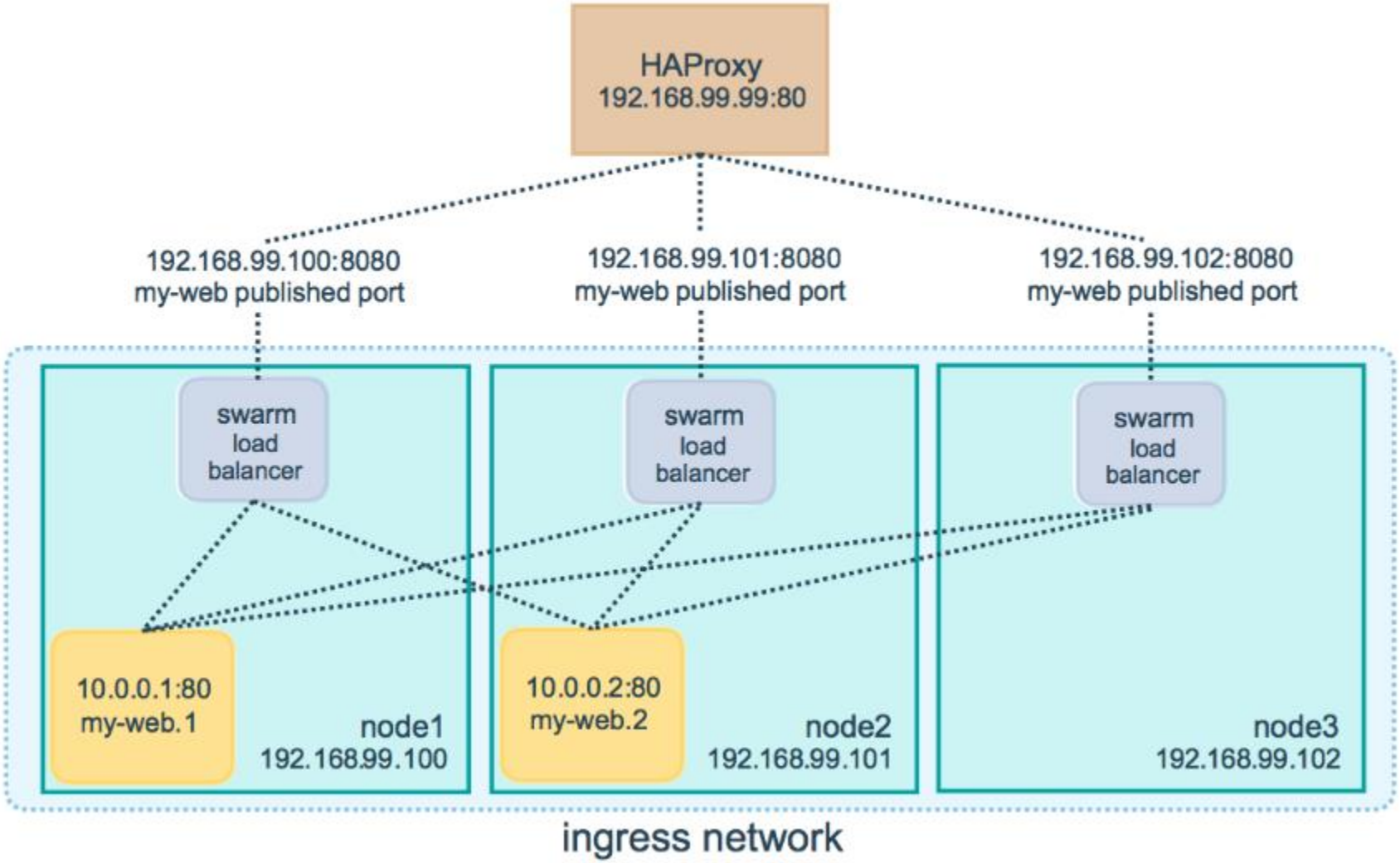


Modo Ingress

```
$ docker service create -d --name svc1 --network overnet \
  --publish published=5000,target=80 nginx
```



Load Balancer externo



Carpeta container_orch_labs:

Lab 1: Docker Swarm

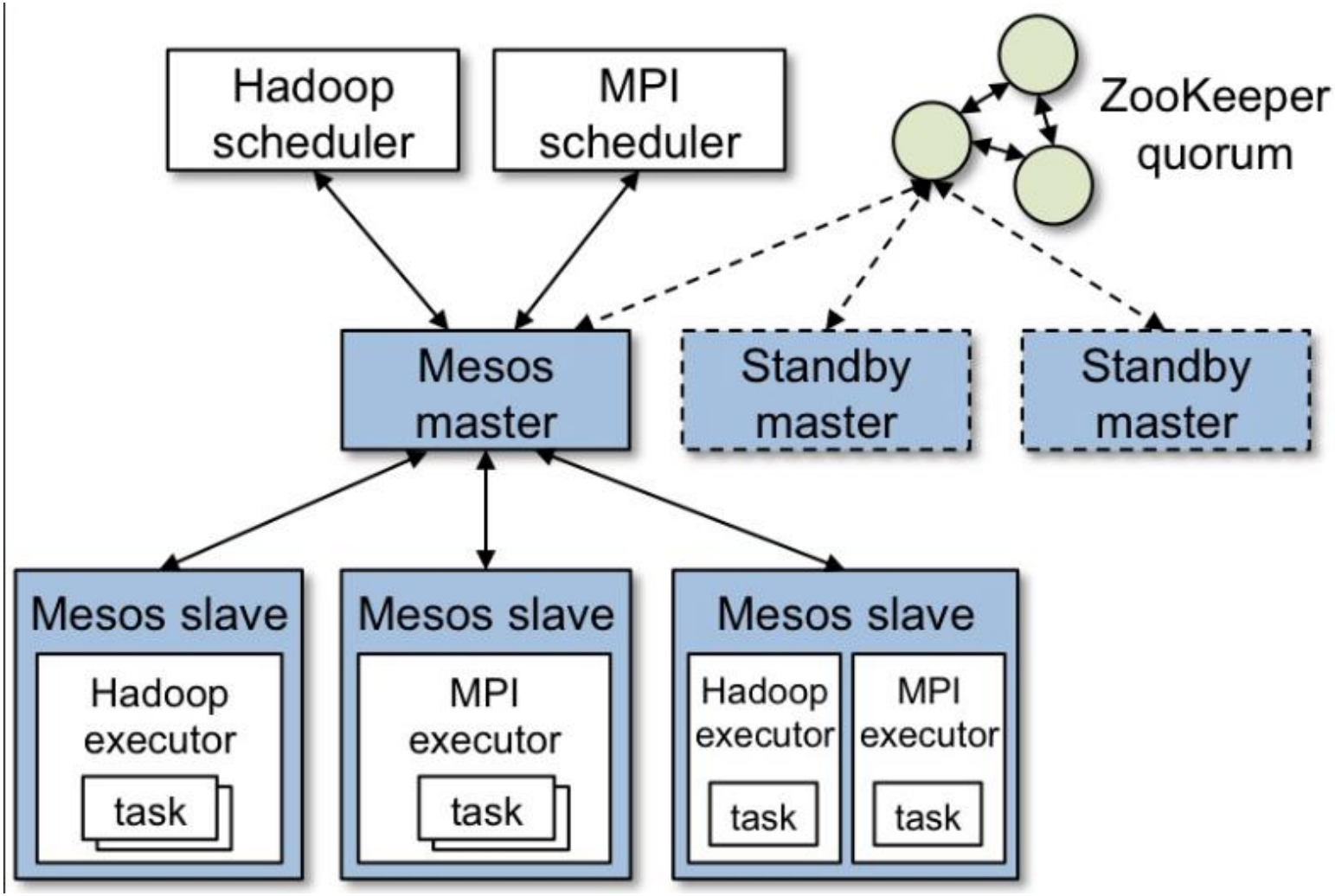
DC/OS

&

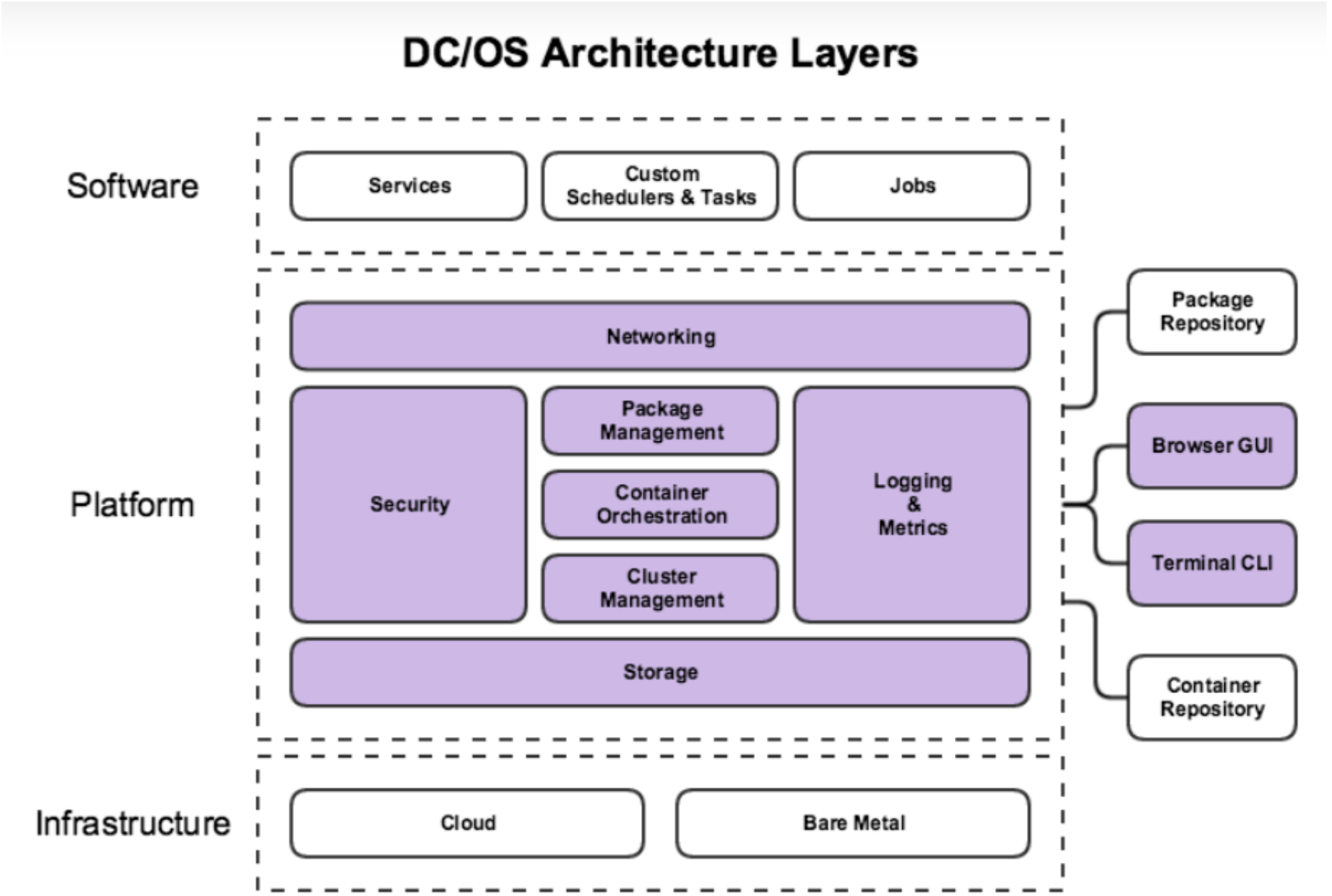
MESOS

DC/OS: Data Center OS

Basado en Apache Mesos.



DC/OS Enterprise



```
{
    "id": "/java-spring-docker",
    "cpus": 0.25,
    "mem": 275,
    "instances": 1,
    "ports": [0],
    "cmd": "java -Xmx256m -jar gs-spring-boot-0.1.0.jar --
server.port=$PORT0 --endpoints.shutdown.enabled=true",
    "labels": {
        "HAPROXY_GROUP": "external",
        "HAPROXY_0_VHOST": "spring.acme.org"
    },
    "constraints": [

    ],
    "healthChecks": [
        {
            "protocol": "HTTP",
            "portIndex": 0,
            "path": "/",
            "gracePeriodSeconds": 300,
            "intervalSeconds": 10,
            "maxConsecutiveFailures": 2,
            "timeoutSeconds": 30
        }
    ],
    "container": {
        "type": "DOCKER",
        "docker": {
            "image": "stathyinc/java-spring:latest",
            "network": "BRIDGE"
        }
    }
}
```

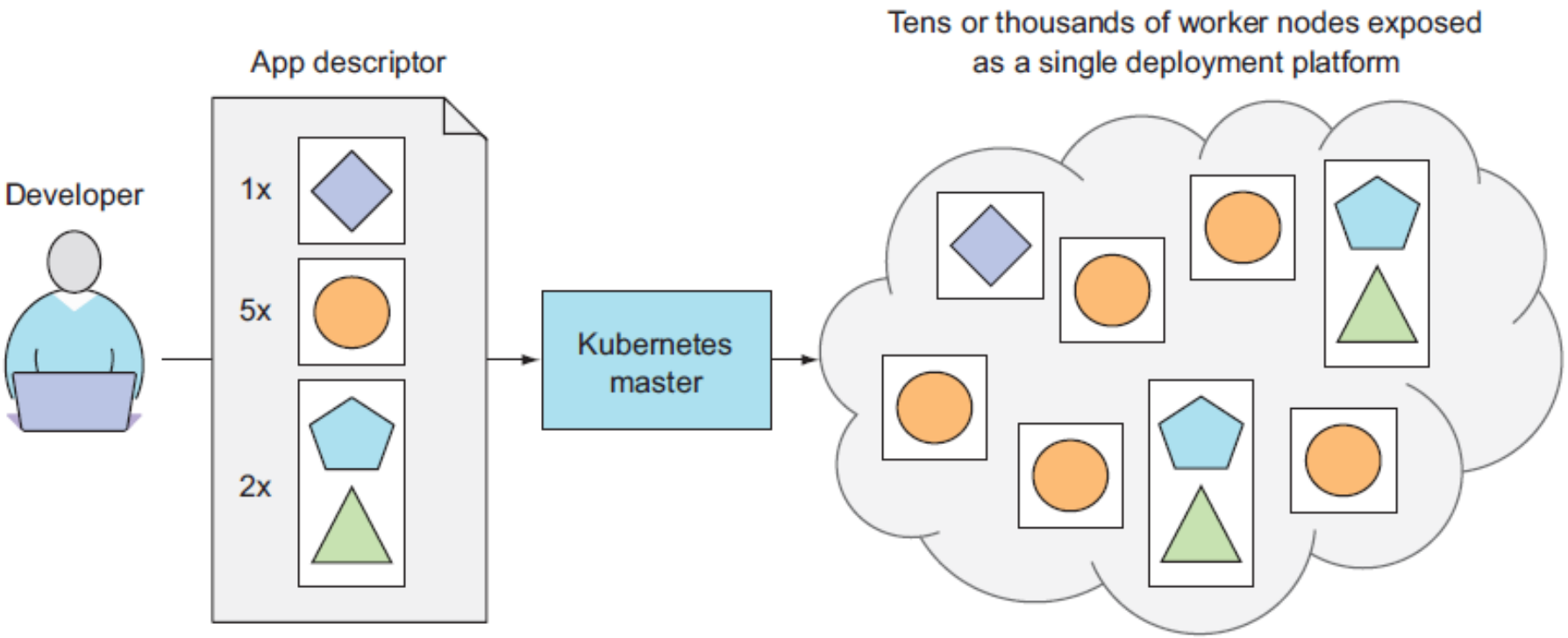
Deploy

Fast Data con SMACK

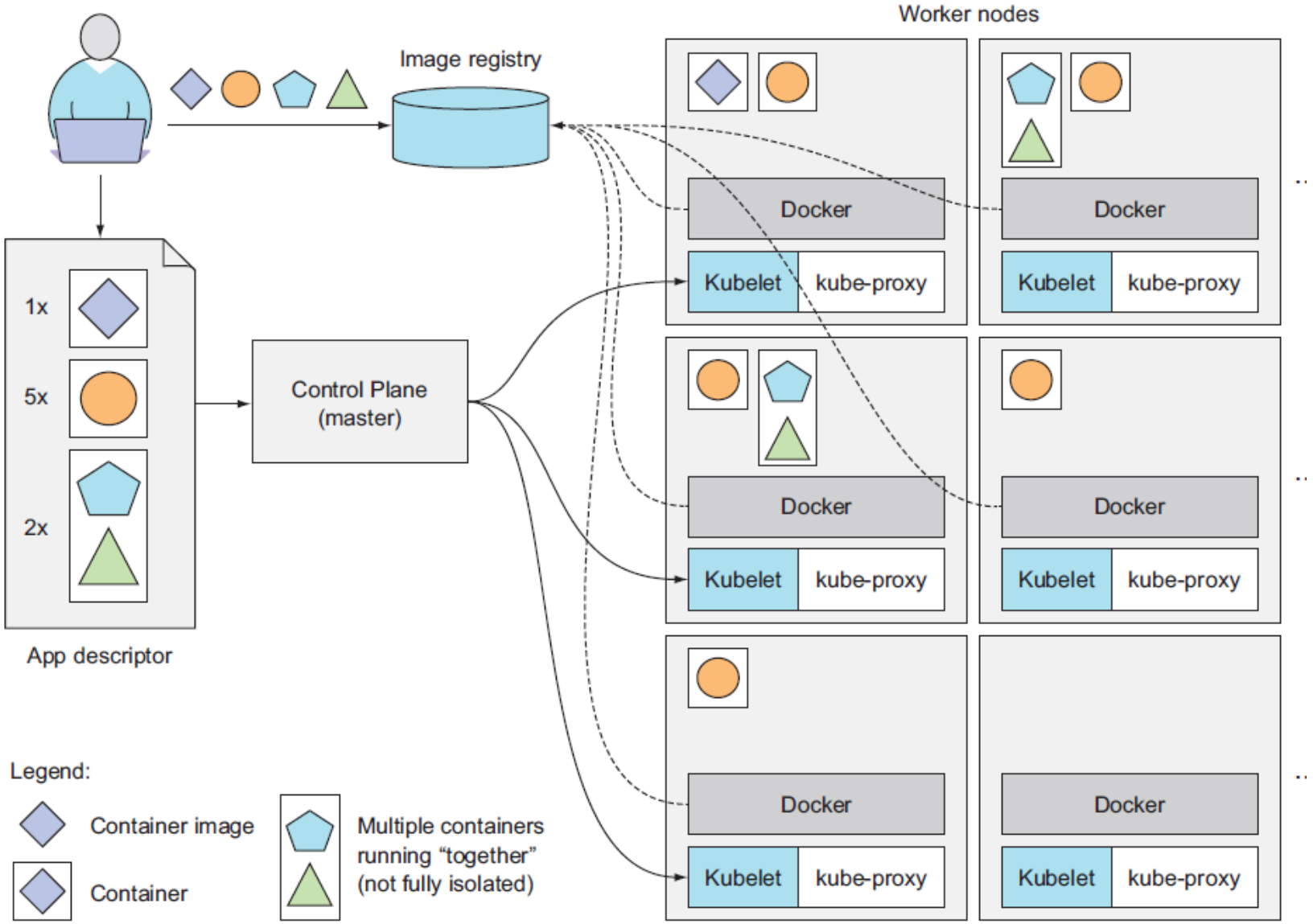
Lab 2: DC/OS Mesos y Marathom

kubernetes

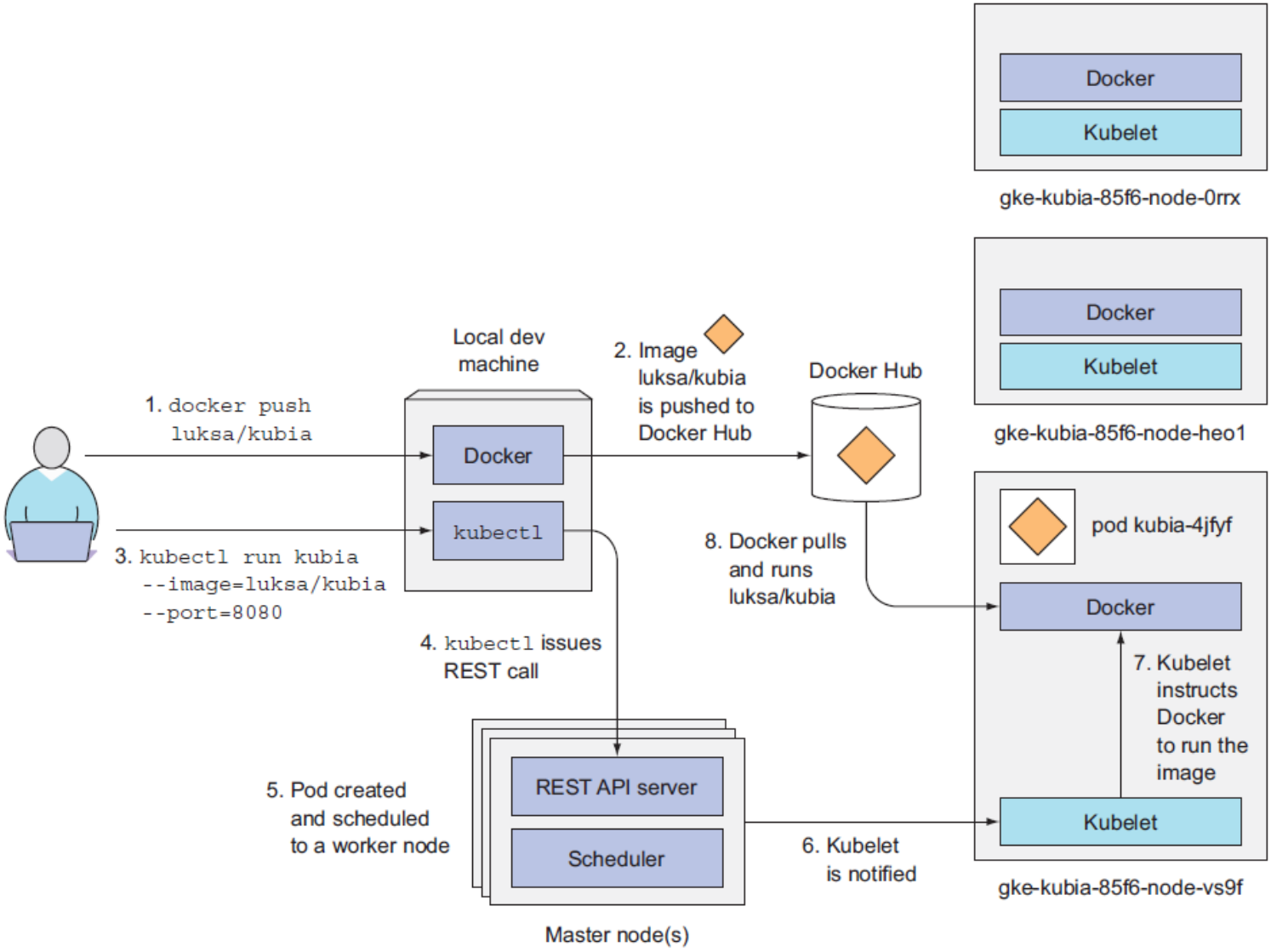
Kubernetes


Tens or thousands of worker nodes exposed as a single deployment platform
App descriptor
Developer
1x
5x
2x
Kubernetes master

Kubernetes: Arquitectura

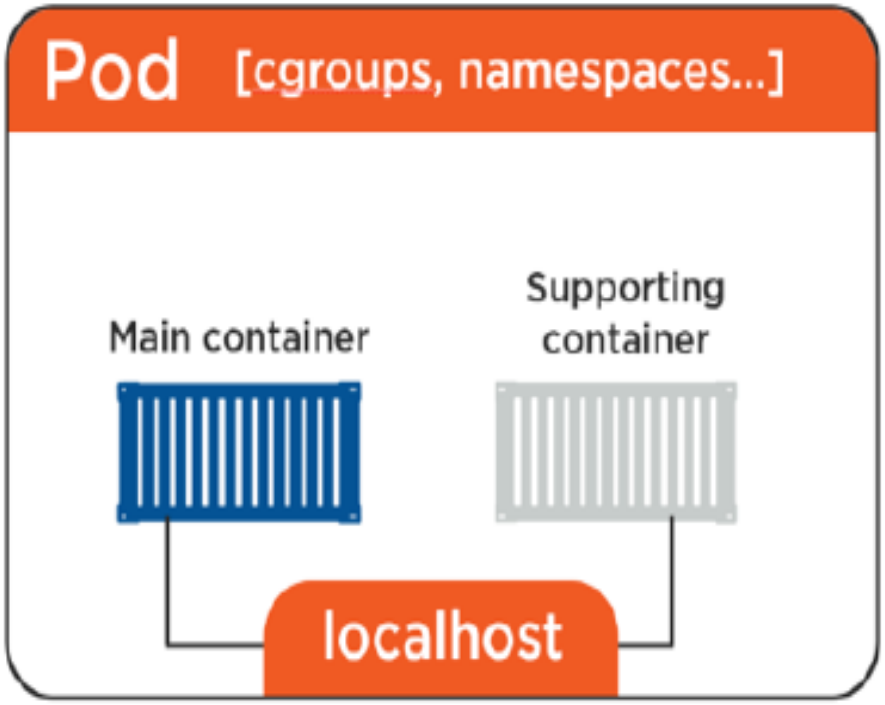


Kubernetes: Arquitectura

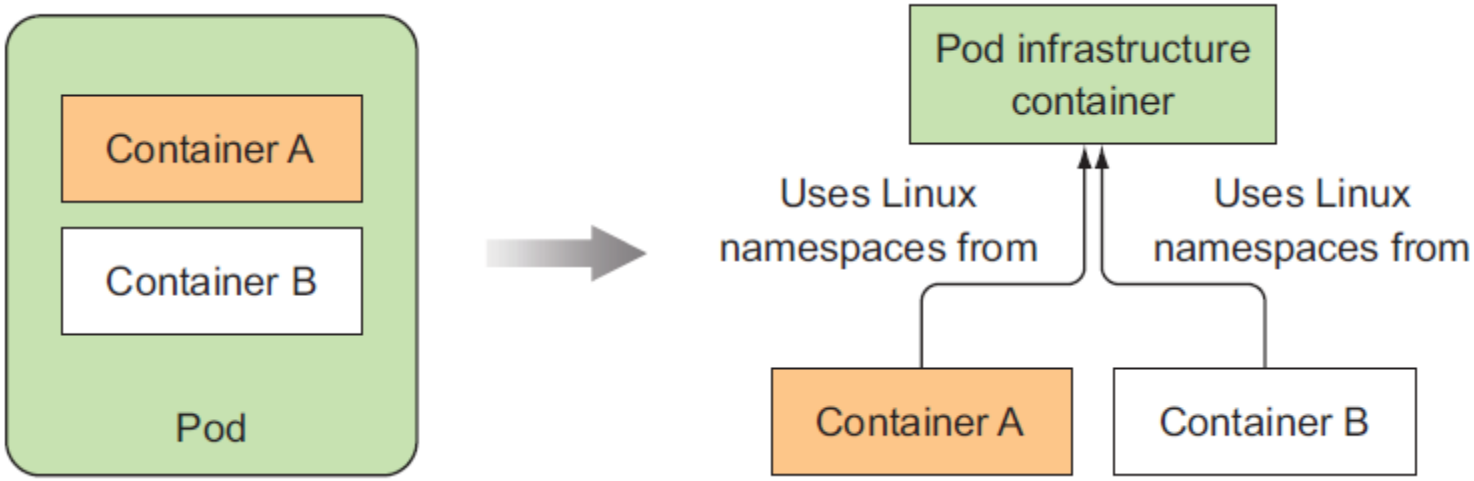


POD

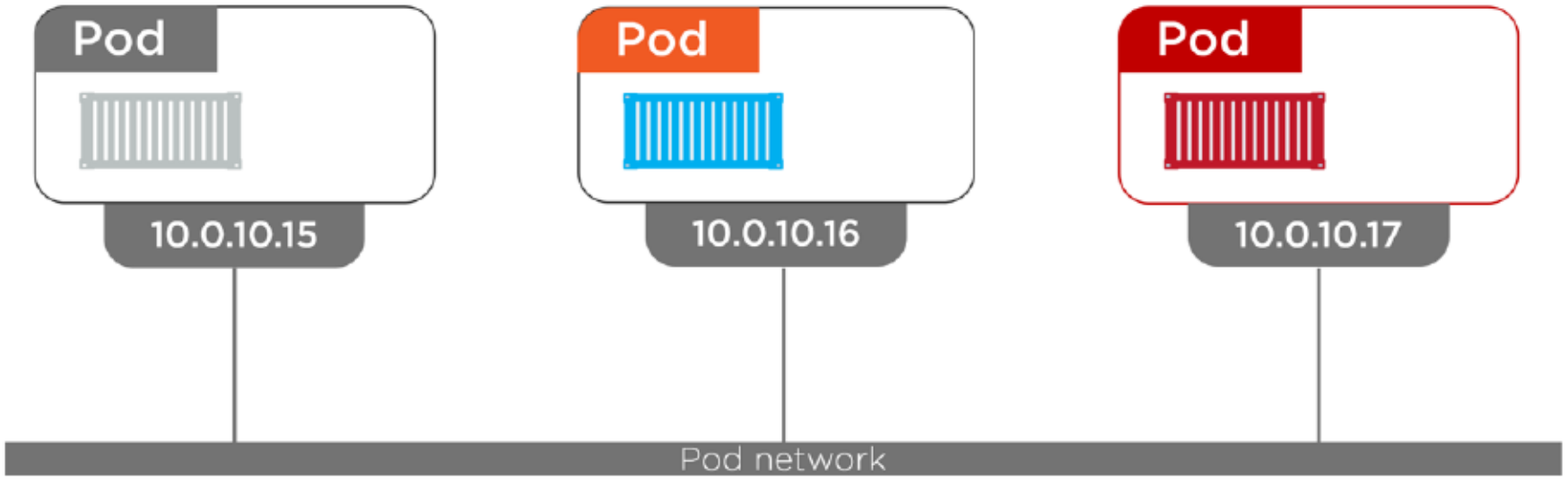
Mínima unidad de despliegue en Kubernetes.
Puede contener más de un Container.



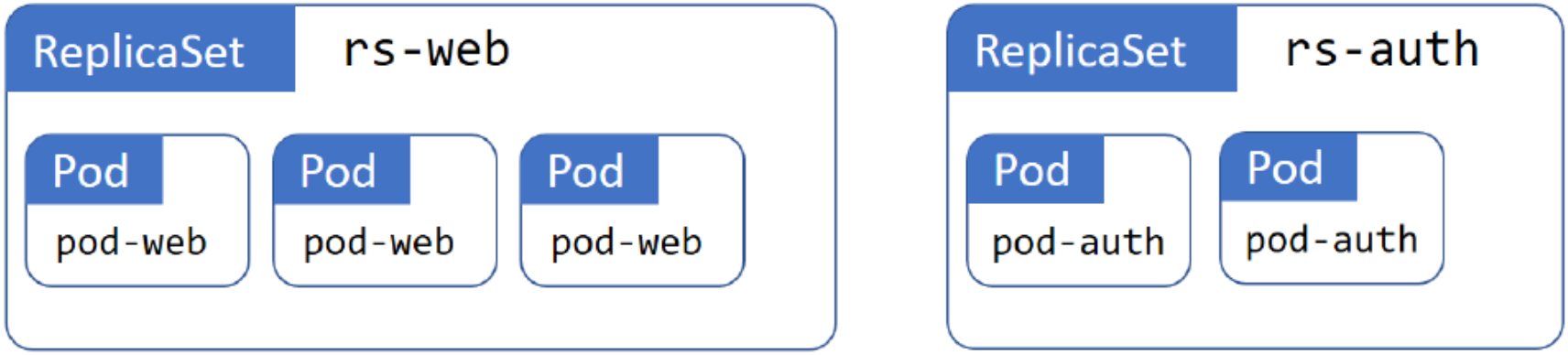
POD


Container A
Container B
Pod
Pod infrastructure container
Uses Linux namespaces from
Uses Linux namespaces from
Container A
Container B

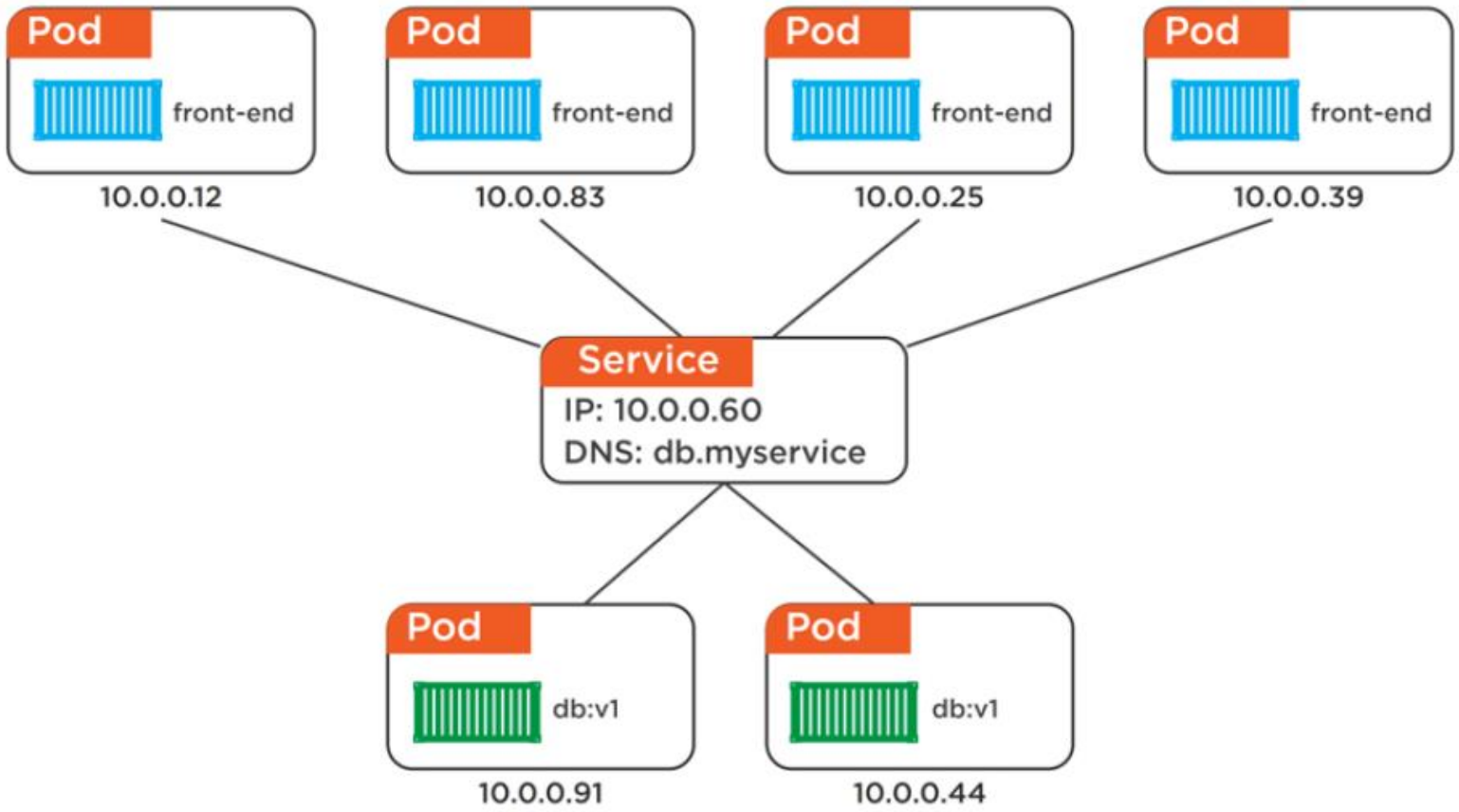
PODs



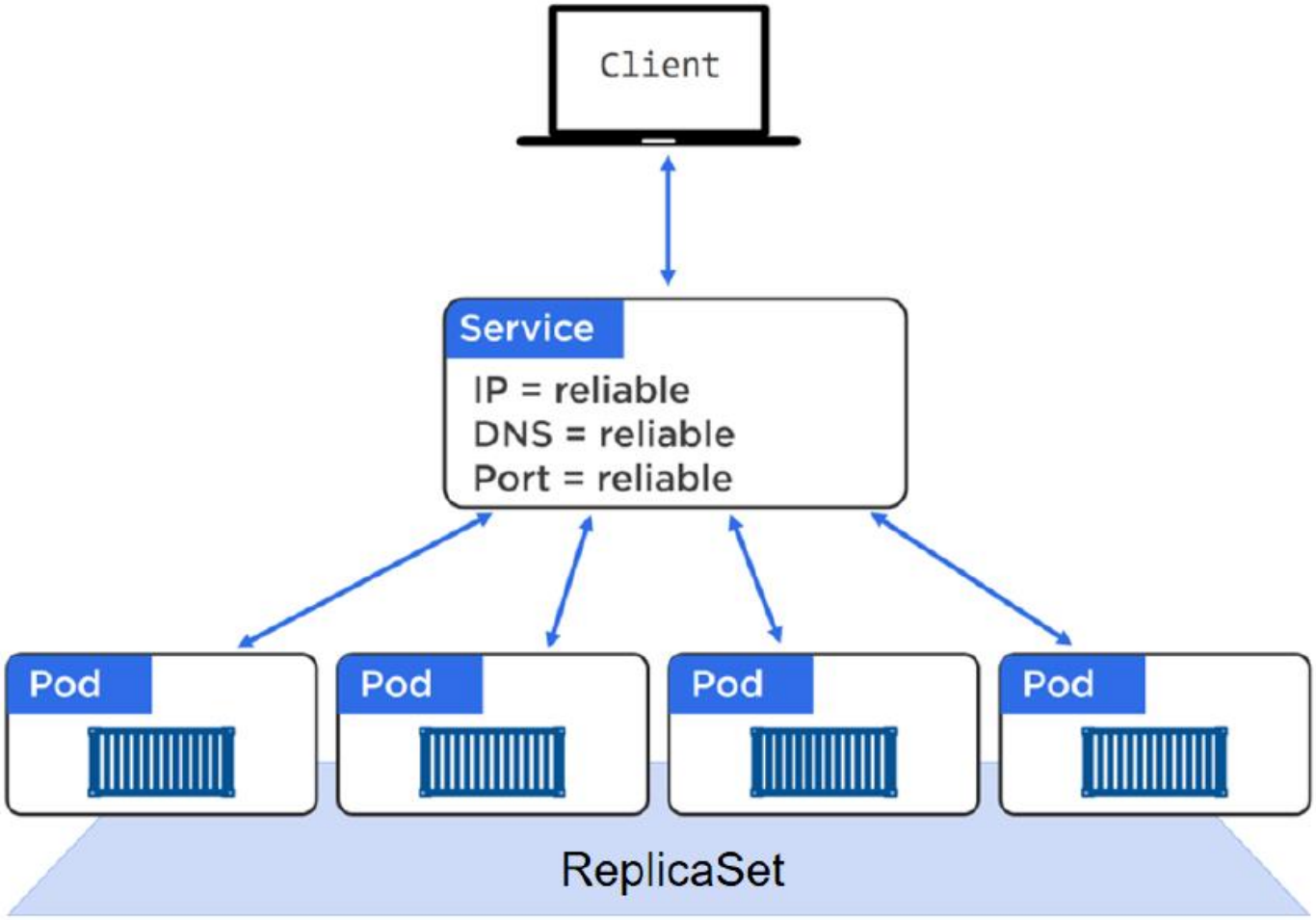
Replica Sets



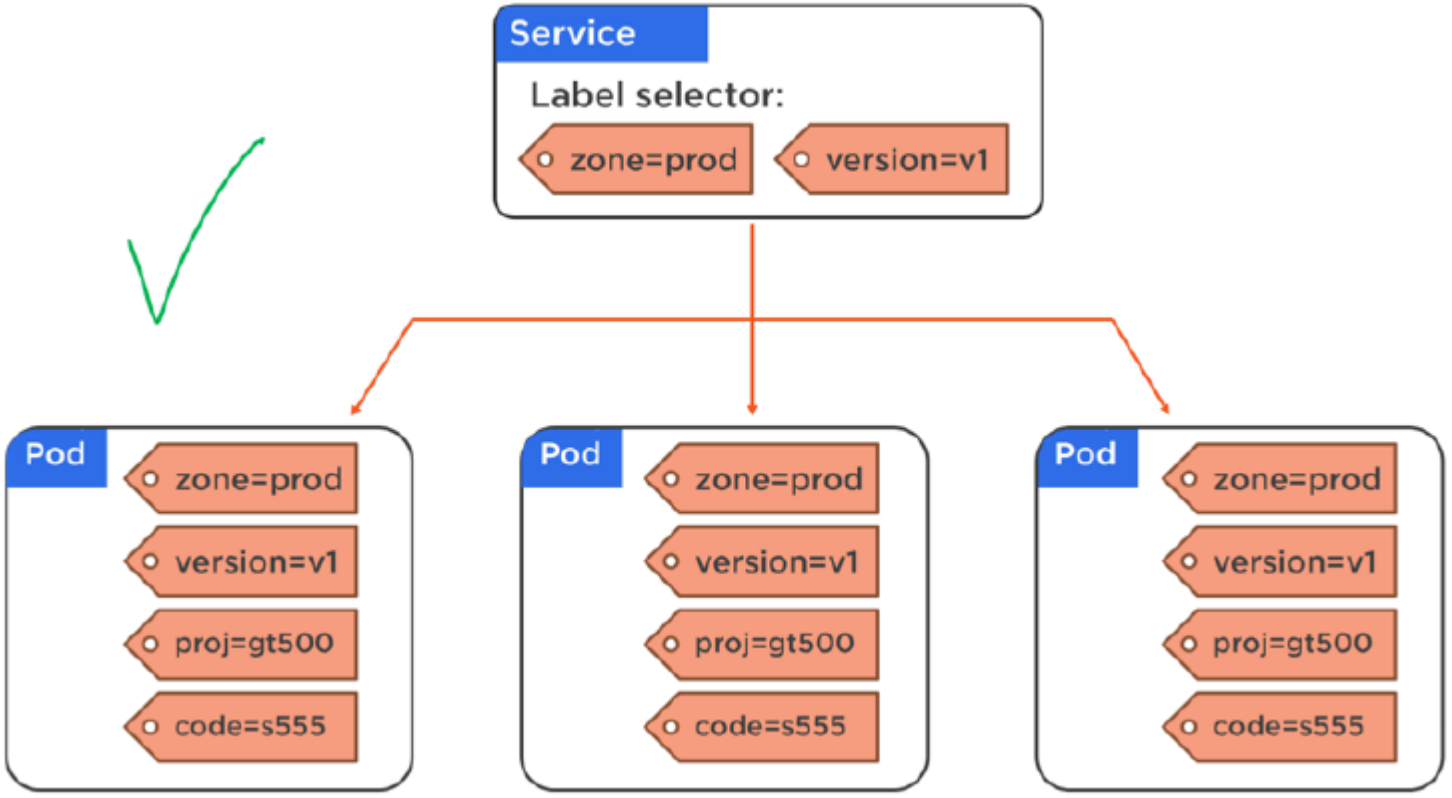
Services


Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.91
Pod
db:v1
10.0.0.44

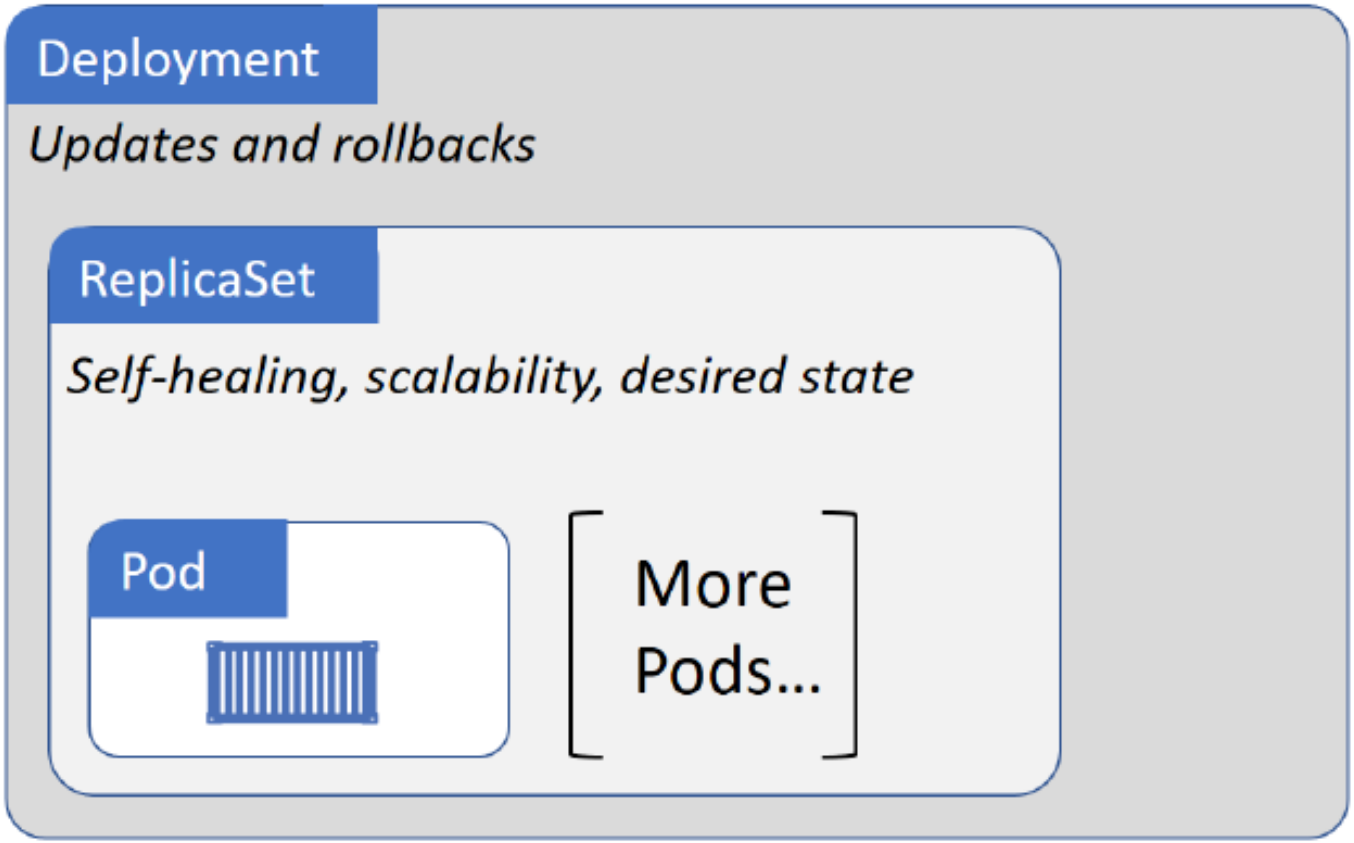
Services


Client
Service
IP = reliable
DNS = reliable
Port = reliable
Pod
Pod
Pod
Pod
ReplicaSet

Services: Uso de Labels



Deployments




```
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod
  labels:
    app: hello-world
spec:
  containers:
  - name: hello-world
    image: diegochavezcarro/hellonodedocker:latest
    ports:
    - containerPort: 8080
```

```
apiVersion: apps/v1beta2
kind: ReplicaSet
metadata:
  name: hello-rs
spec:
  replicas: 10
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
  spec:
    containers:
    - name: hello-world
      image: diegochavezcarro/hellonodedocker:latest
      ports:
      - containerPort: 8080
```

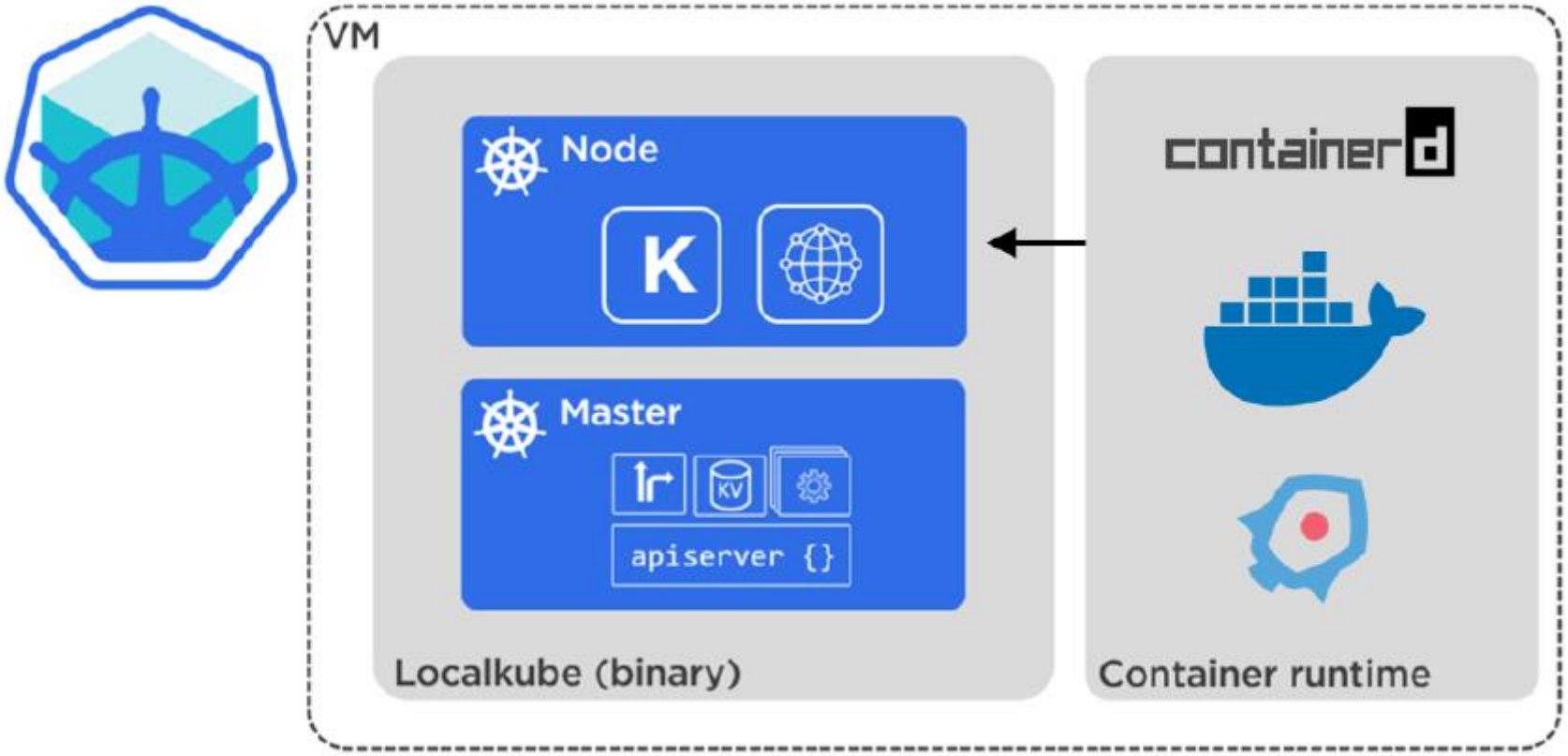
```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: hello-deploy
spec:
  replicas: 10
  selector:
    matchLabels:
      app: hello-world
  minReadySeconds: 10
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - name: hello-pod
        image: diegochavezcarro/hellonodedocker:latest
        ports:
        - containerPort: 8080
```

Deployment

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: hello-svc
  labels:
    app: hello-world
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 30001
    protocol: TCP
  selector:
    app: hello-world
```

Minikube


VM
Node
K
Master
KV
apiserver {}
Localkube (binary)
containerd
Container runtime

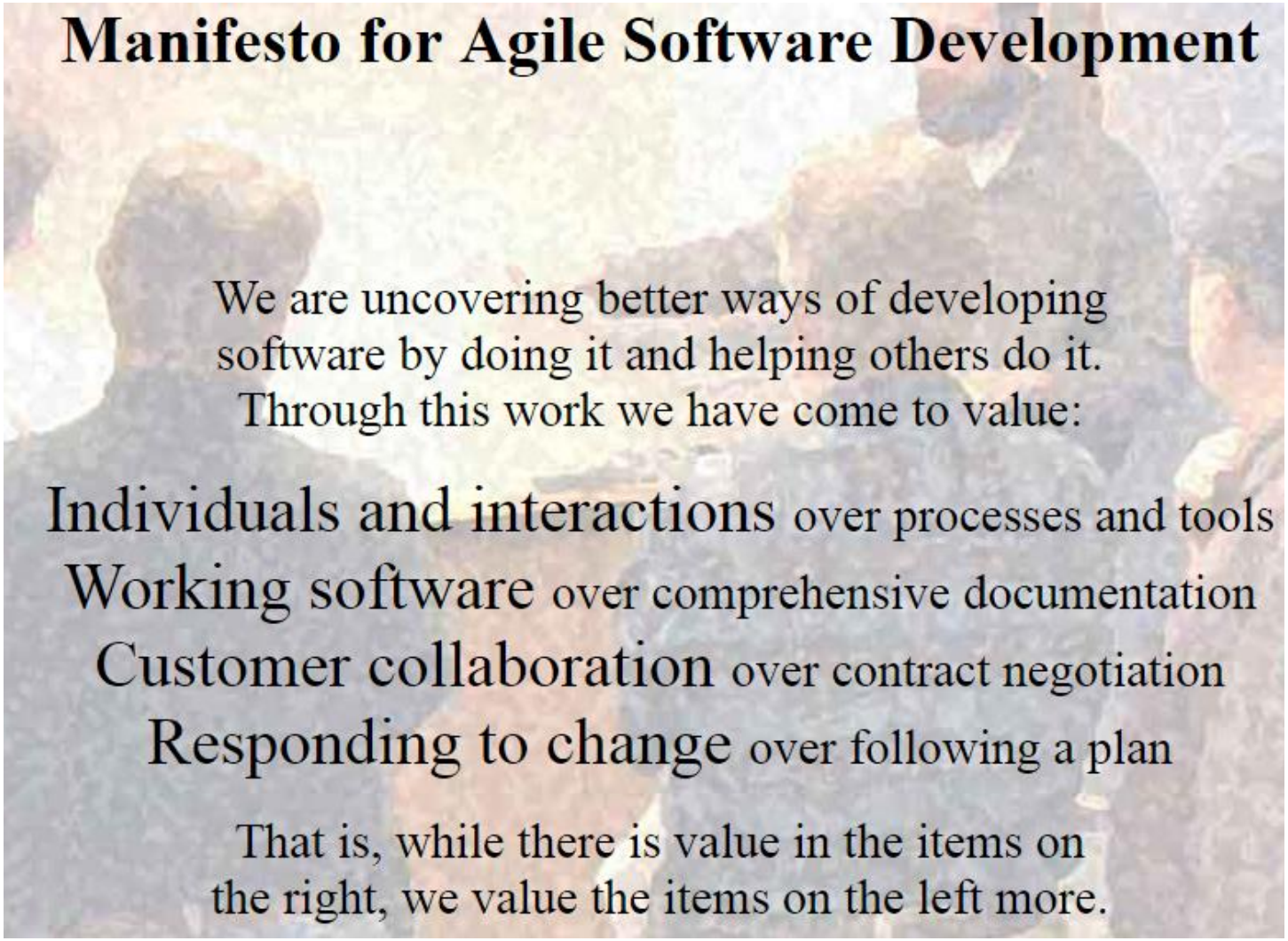
Lab 3: Kubernetes Deploy Modo Imperativo

Lab 4: Kubernetes Deploy Modo Declarativo

Lab 5: Kubernetes Deploy con Docker for Windows

Devops =
Agile + CI +CD

Agile



1. http://agilemanifesto.org/

Agile

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Agile

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

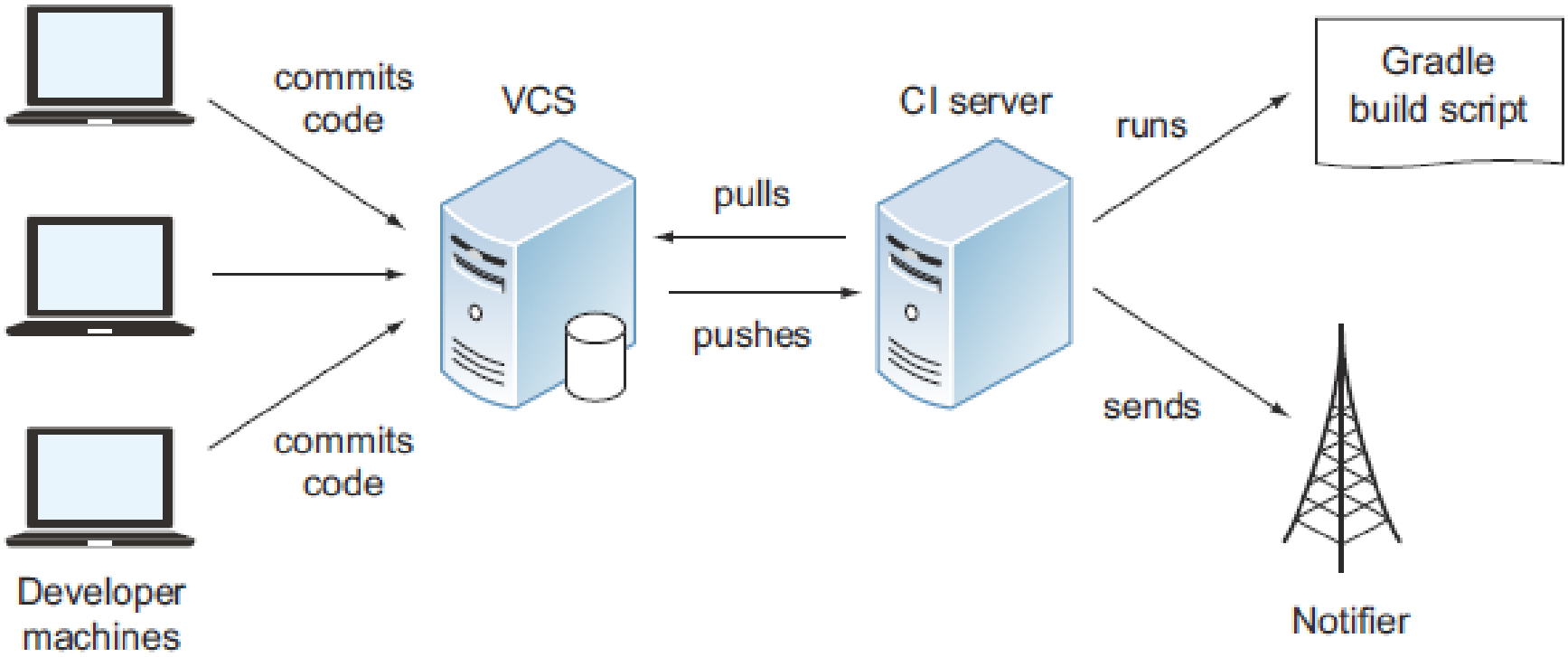
Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Integración Continua y Delivery Continuo

- Realizar integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes.

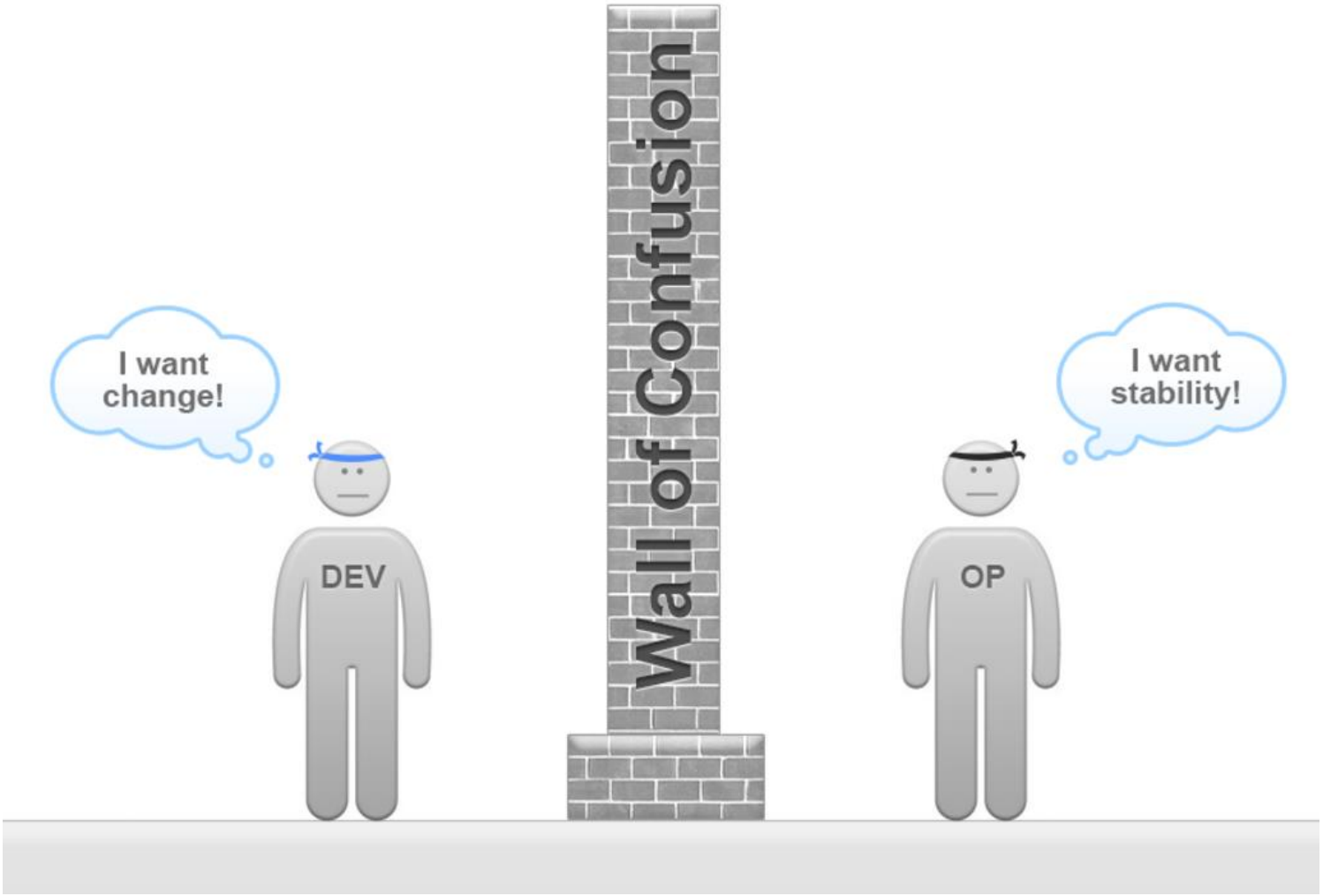


DevOps

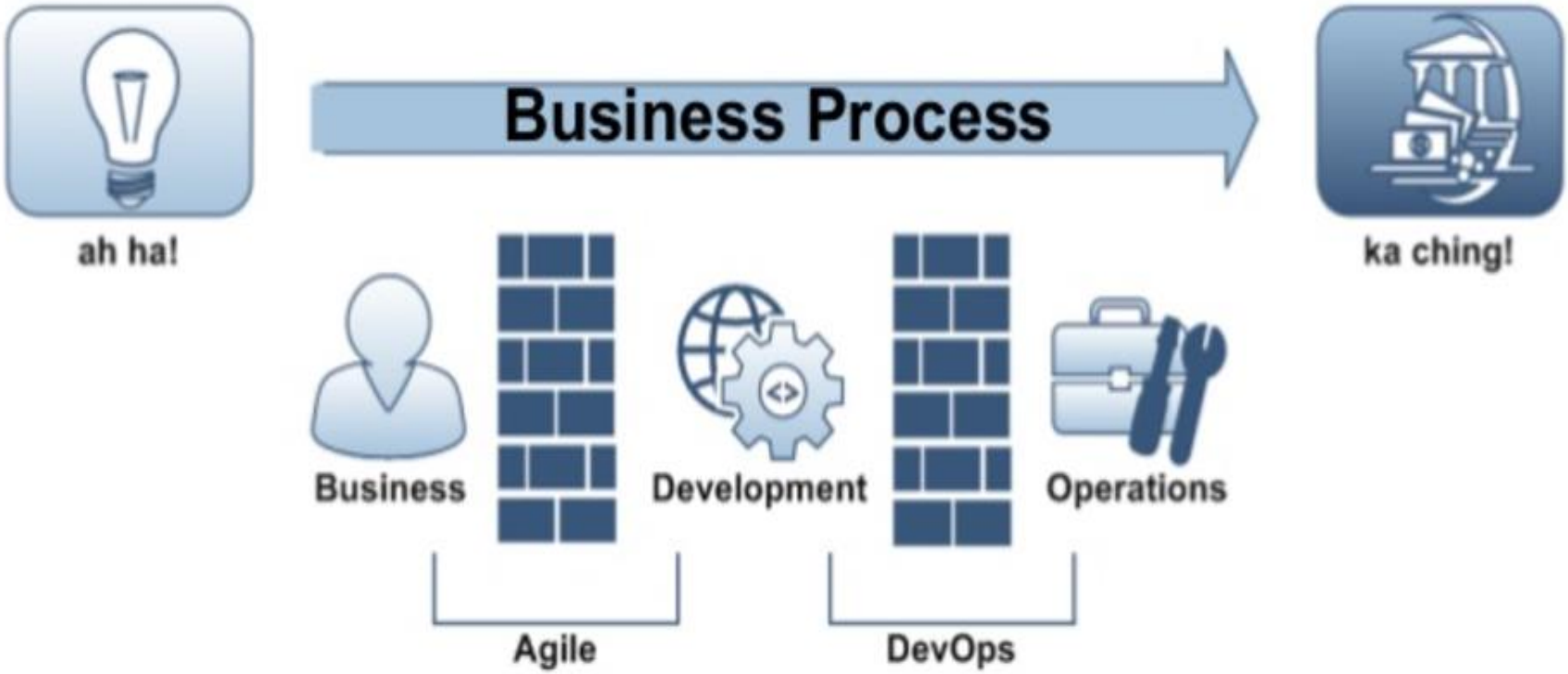
- DevOps es un acrónimo inglés de development (desarrollo) y operations (operaciones), que se refiere a una cultura o movimiento que se centra en la comunicación, colaboración e integración entre desarrolladores de software y los profesionales en las tecnologías de la información (IT).

- Permite automatizar el proceso de entrega del software y los cambios en la infraestructura. Su objetivo es ayudar a crear un entorno donde la construcción, prueba y lanzamiento de un software pueda ser más rápido y con mayor fiabilidad.

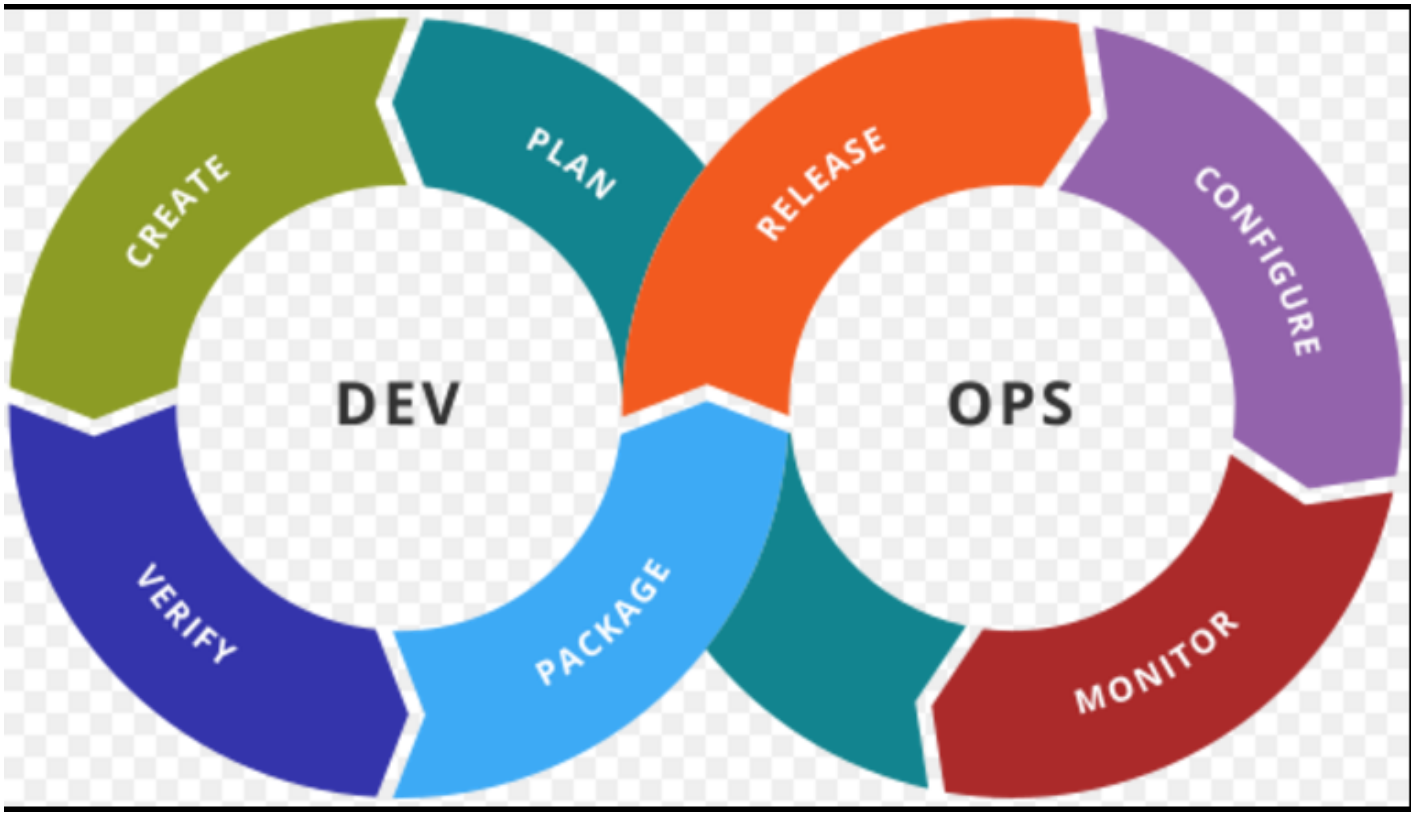
Wall of Confusion



IT Alignment and Business Agility



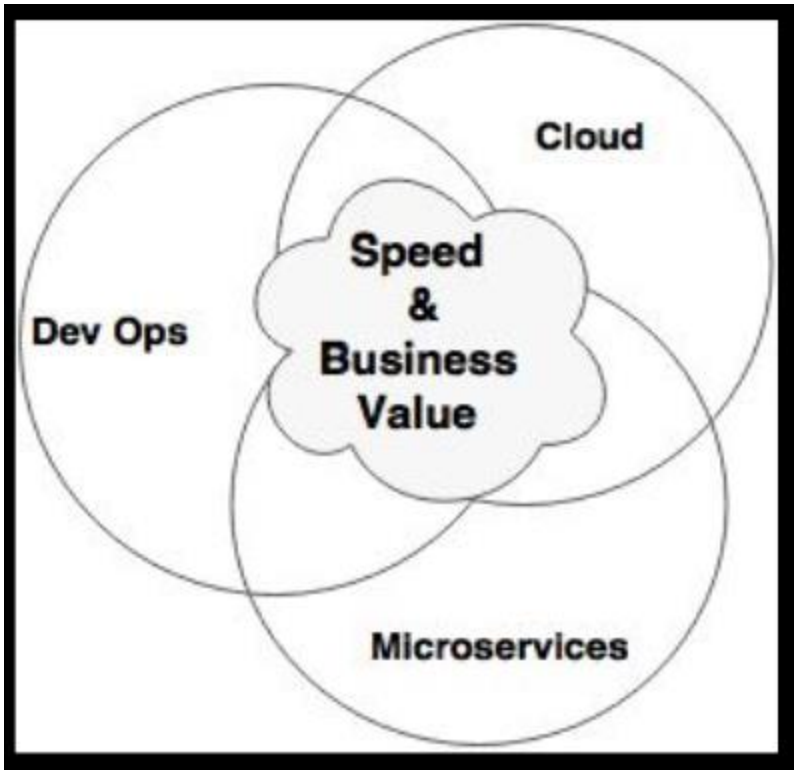
Proceso de Desarrollo



1. https://less.works/less/structure/feature-teams.html
2. https://martinfowler.com/bliki/BusinessCapabilityCentric.html
3. https://martinfowler.com/bliki/AlignmentMap.html

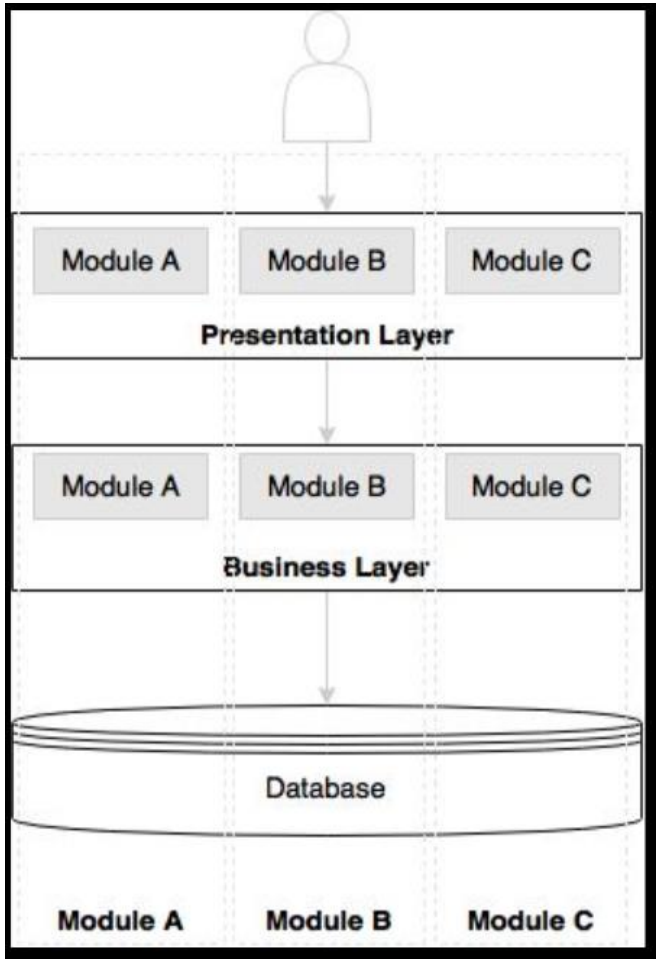
Devops + Microservicios + Cloud

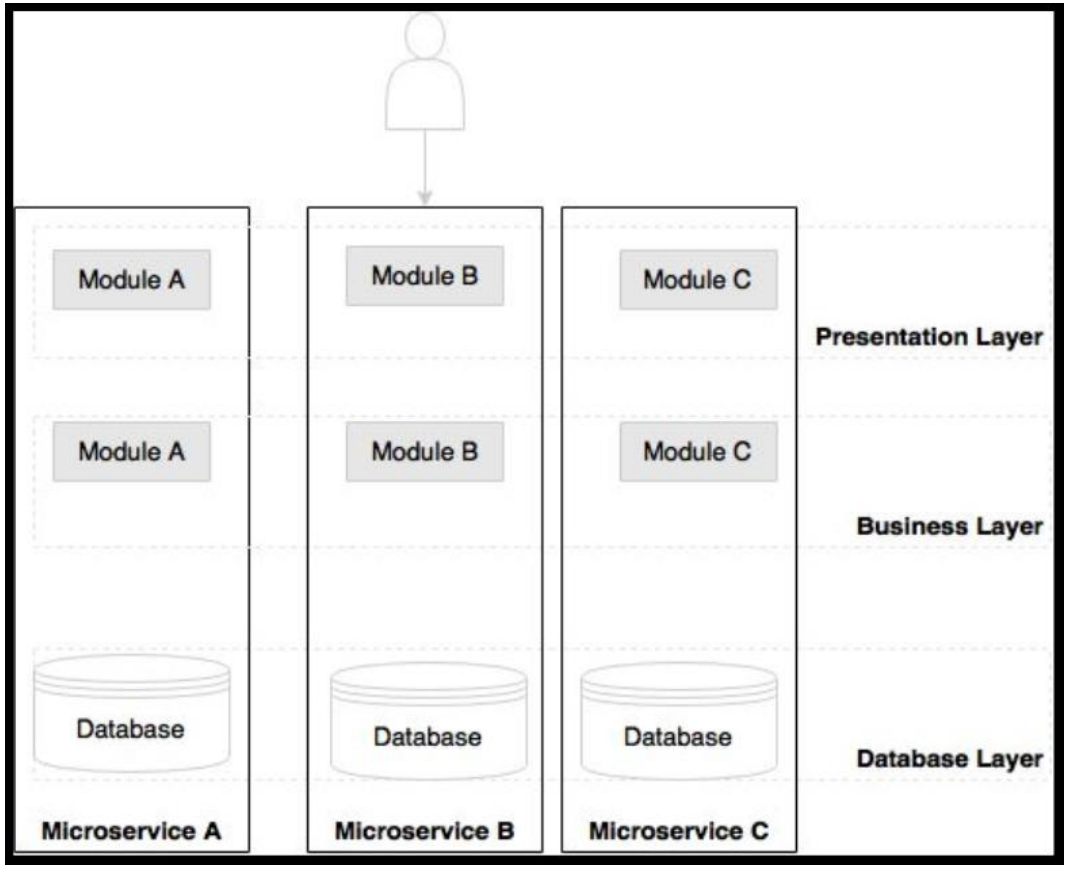
DevOp, Microservicios y Cloud



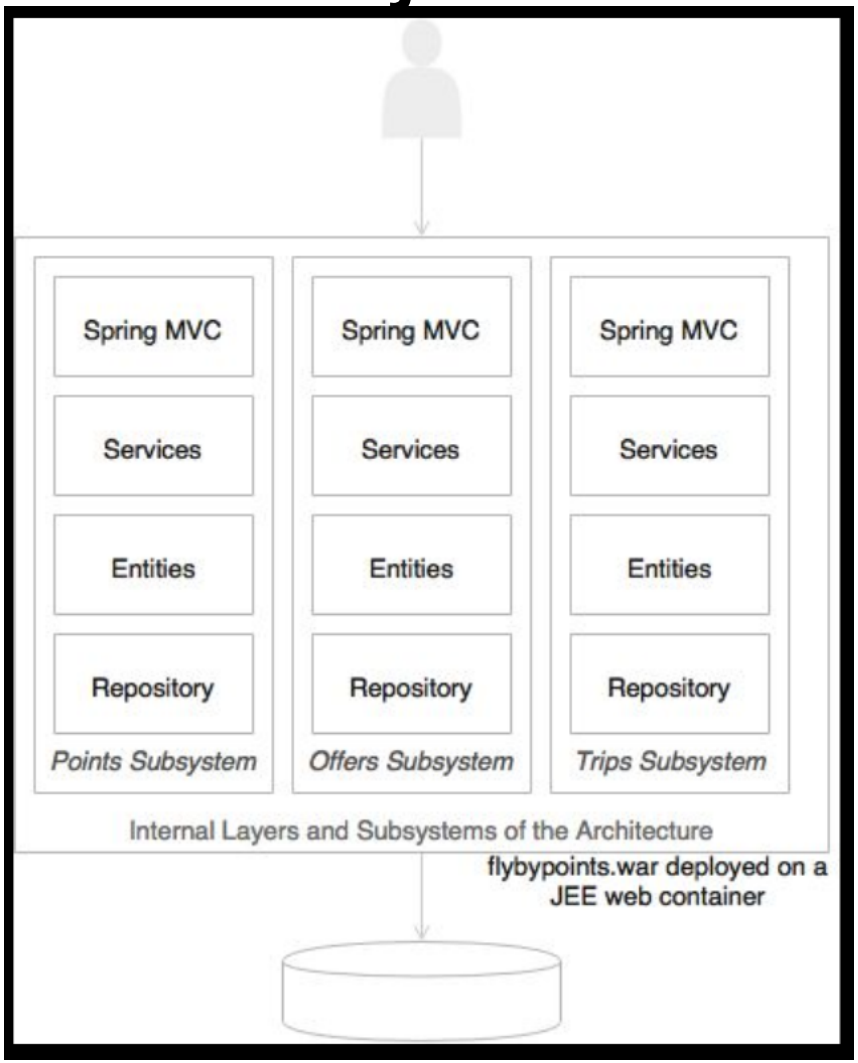
1. https://12factor.net/

Aplicaciones Monolíticas vs Microservicios

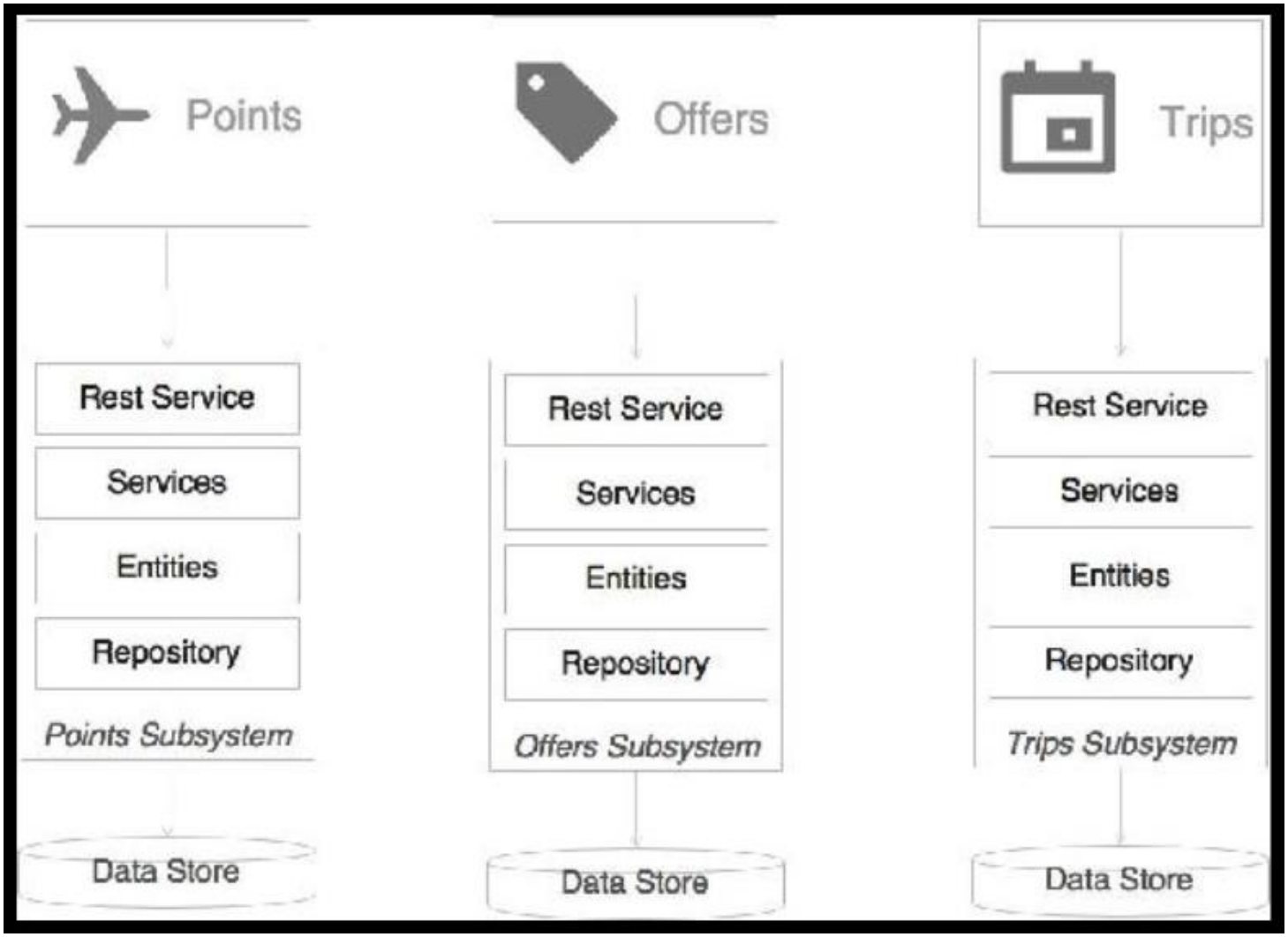




Ejemplo de App de Puntos para Viajes



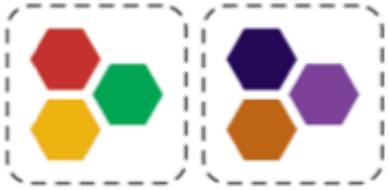
Decomposición usando Headless Microservices



Escalamiento Granular

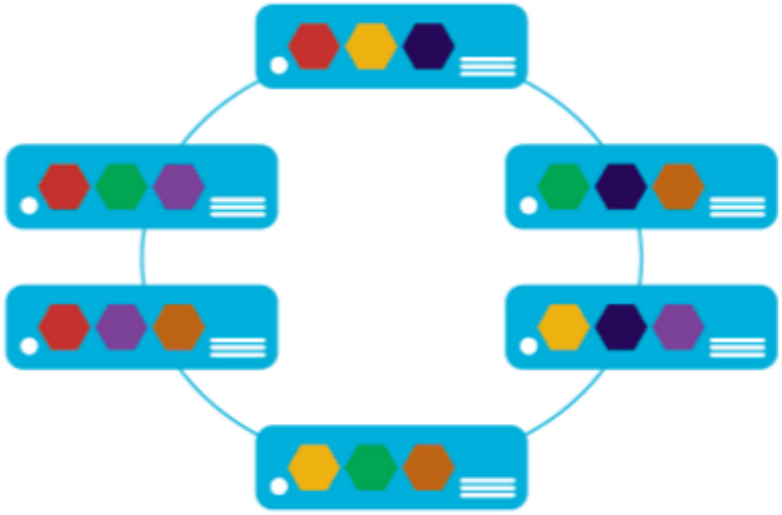
Microservices Approach

App 1 App 2



A microservice approach segregates functionality into small autonomous services.

And scales out by **deploying independently** and replicating these services across servers/VMs/containers.



VS.

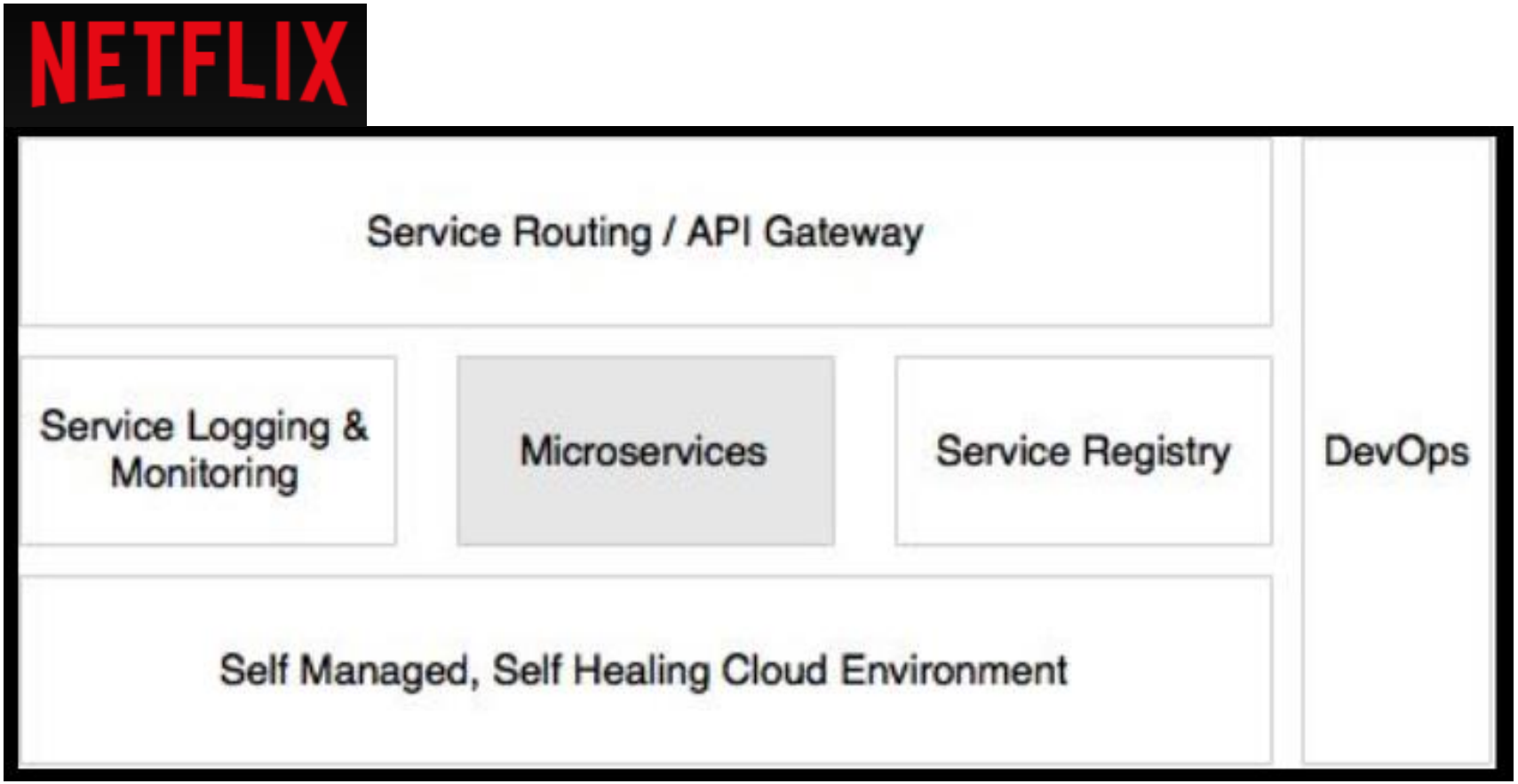
Traditional Approach

App 1

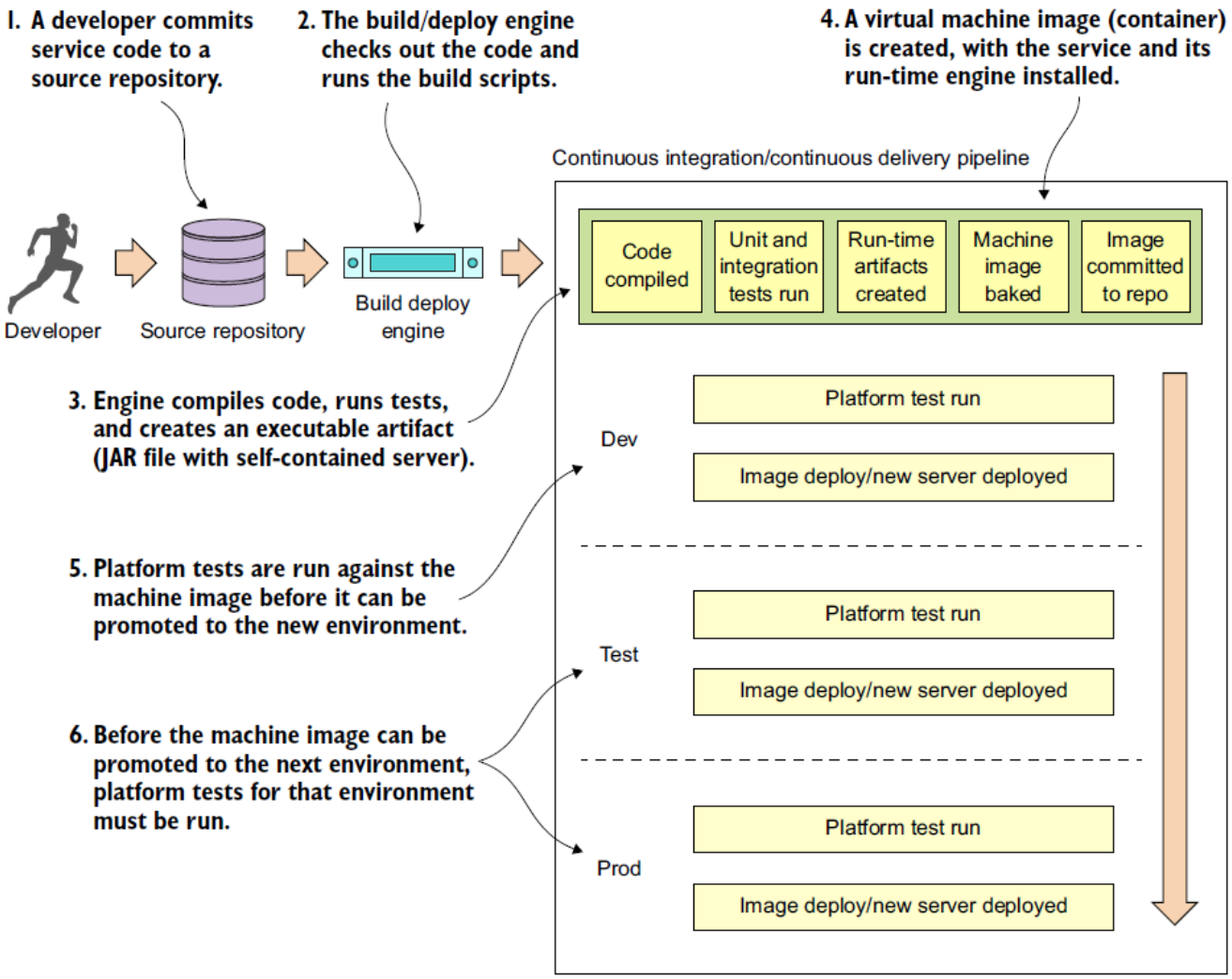
A traditional application (Web app or large service) usually has most of its functionality within a single process (usually internally layered, though).

And scales by cloning the whole app on multiple servers/VMs/containers.

Arquitectura en empresas nativas digitales

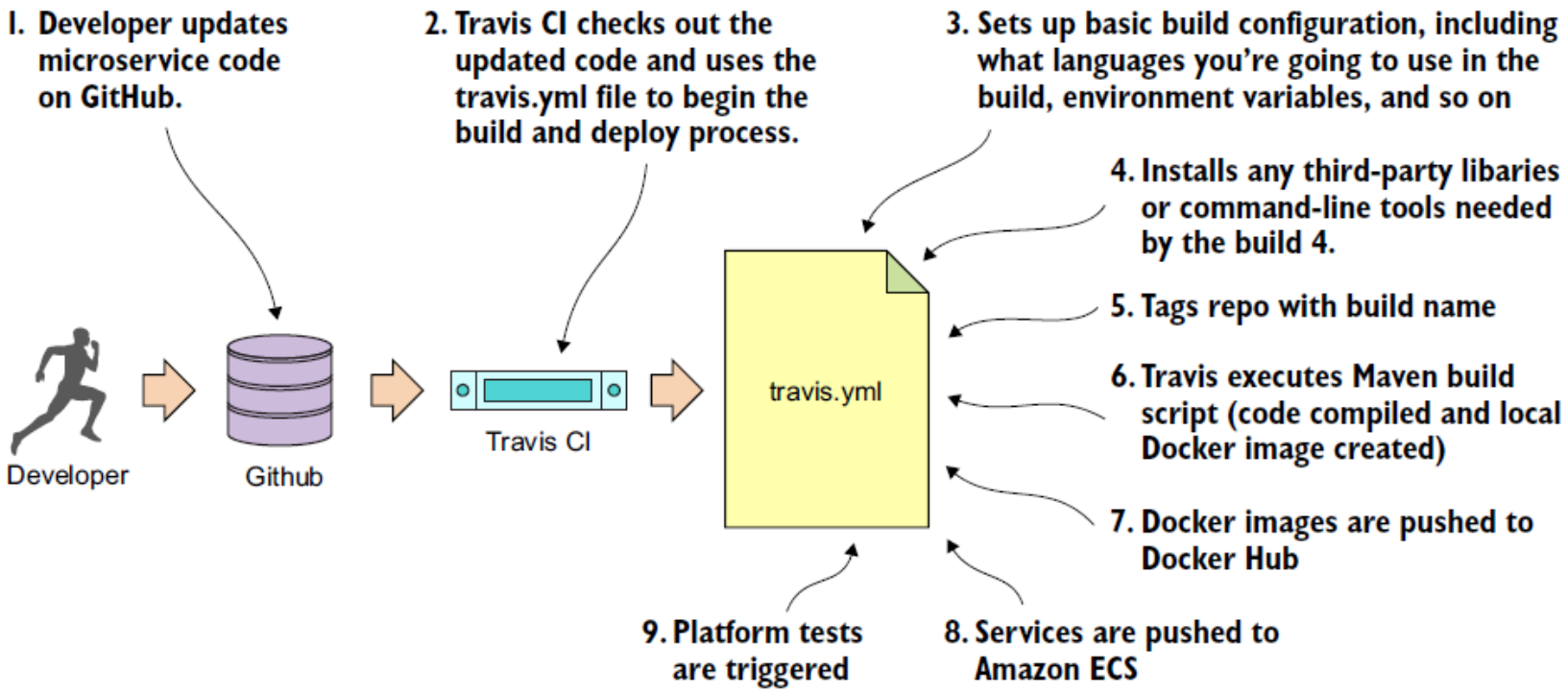


DevOps, Microservicios y Cloud



Trazabilidad

- Tagging del Repositorio y de la Imagen



Preguntas?

Diego Chávez

diegochavezcarro@gmail.com